Desenvolupament d'una aplicació per monitorar l'exercici físic en carrera.

RunGo

Albert Aige Cortasa

Directora TFG: Montse Sendin Veloso

Juliol de 2014

Índex

1 Introducció

La plataforma Android ha estat creixent molt durant els últims anys, ha passat de tenir un 3,9% de vendes l'any 2009 a l'any 2013 tenir un 78,4% del total d'SmartPhones venuts. Amb aquestes dades es pot veure el gran mercat que te la plataforma Android.

En aquest treball he realitzat una aplicació que permet monitorar diferents paràmetres d'una persona que esta fen exercici físic, principalment corrent. Recollirà paràmetres com la distancia recorreguda, la ruta, etc.. Tot això es farà utilitzant la nova versió d'Android 4.4 Kit Kat la qual incorpora dos nous sensors, l'step counter i step detector, amb l'objectiu de contar les passes al córrer o caminar, també s'utilitzarà l'eina de Google Maps i el sensor GPS per obtenir la ruta i mostrar-la. Tota la informació recollida s'enviarà en un Cloud, concretament en el de Google, per a la posterior consulta d'aquestes.

L'usuari final serà qualsevol persona que vulgui monitorar qualsevol exercici físic, preferiblement en els quals es faci un recorregut en ruta. Aquesta aplicació li permetrà al usuari supervisar el seu esforç i observar les millores en el seu estat duran el temps.

1.1 Objectius del treball

Cada vegada més les plataformes mòbils estan guanyat protagonisme dins el mercat de la informàtica. Per això he trobat oportú centrar el meu treball en aquestes plataformes, concretament en Android, ja que tenen un futur molt prometedor.

Conèixer aquesta plataforma sempre ha sigut un dels meus objectius desde que vaig començar a estudiar informàtica, però per la falta de temps i coneixements de la programació no vaig aprofundir-hi.

Finalment duent a terme aquest treball podré construir una aplicació que solucioni algun problema en la vida real i augmentar els meus coneixements sobre aquesta plataforma.

2 Viabilitat i planificació

Els últims anys la societat esta prenen més consciència de la seva salut, es a dir, dediquen una part del seu temps lliure a realitzar activitats per a cuidar-se, en el nostre cas volem ajudar a que aquestes persones portin un control fent servir la tecnologia mòbil.

Qualsevol tipus d'usuari que disposi d'un dispositiu mòbil amb una versió Android 4.4 o superior, serà capaç d'utilitzar l'aplicació. Tan si es un corredor amateur o professional podrà gaudir de la l'aplicació.

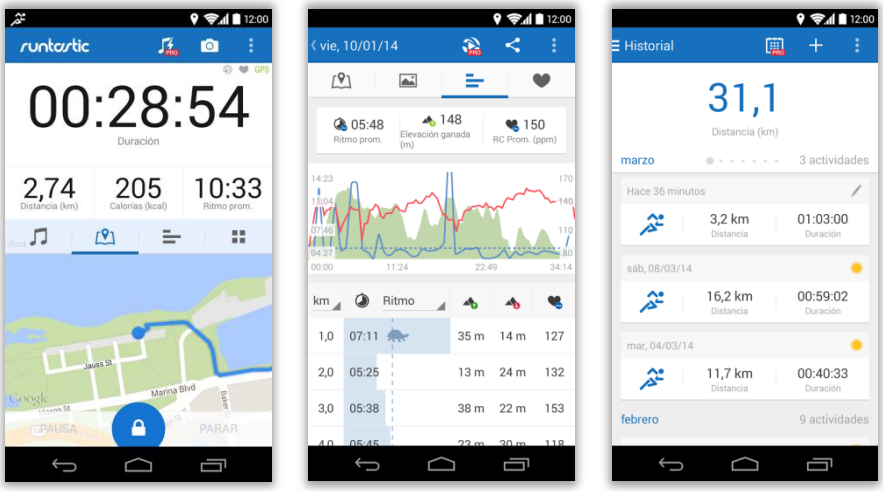
2.1 Estudi de la competència

Ja existeixen aplicacions similars com la que estic creant en aquest projecte. Anem a veure una mica com són:

Runtastic

És una de les aplicacions més conegudes i amb una gran quantitat de funcionalitats en les quals destaquen:

- Mapes d'entrenament en temps real, monitoratge del progrés i històrics de les sessions d'entrenament.
- Gràfiques: elevació, ritme, velocitat, i ritme cardíac durant la ruta.
- Entrenador per veu.
- Reproductor de musica integrat.
- Possibilitat de compartir les activitats a Google+, Facebook i Twitter.

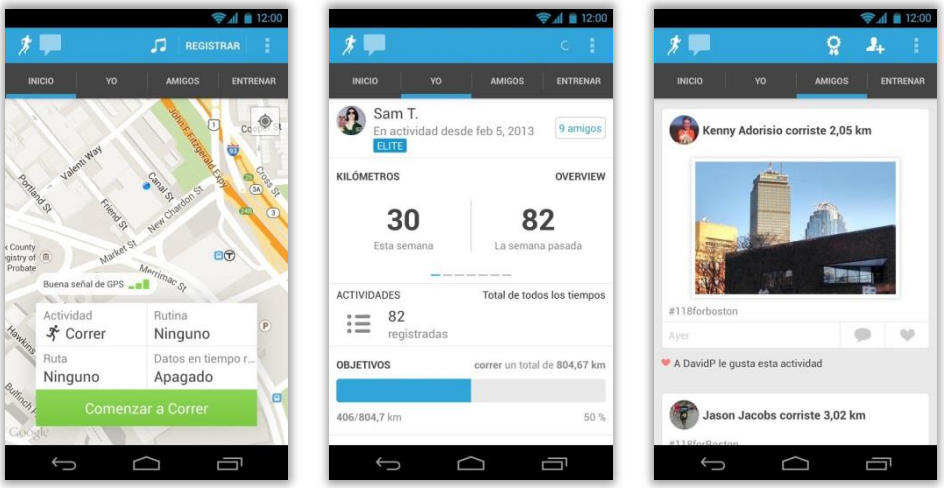


Captures 1. Captures de l'aplicació Runtastic.

RunKeeper

Una altra aplicació que porta bastant temps en el mercat, aquestes són les funcionalitats més destacades:

- Estadístiques detallades del ritme, distancia i temps.
- Obtenció de estadístiques, progrés i entrenament per veu a través dels auriculars.
- Historial detallat de les activitats.
- Mesura els teus progressos per aconseguir els teus objectius i metes.
- Possibilitat de compartir activitats a Facebook i Twitter.



Captures 2. Captures de l'aplicació RunKeeper.

2.2 Anàlisis DAFO

Anàlisis per a estudiar les fortaleses, debilitats, oportunitats i amenaces en la realització del projecte.

	Fortaleses	**Debilitats**
Anàlisis intern	No hi ha límits per les funcionalitats de l'aplicació. Nova versió d'Android amb sensors especialitzats en el moviment de la persona.	El software s'ha de començar de zero.
	Oportunitats	**Amenaces**
Anàlisis extern	El mercat no està saturar pet aquest tipus d'aplicacions.	Ja hi ha algunes aplicacions semblants en el mercat. És complicat que les persones es canviïn a una altra aplicació si la que fan servir satisfà les seves necessitats.

Taules 1. Taula anàlisis DAFO.

2.3 Estimació de costos

Aproximació del cost de realitzar el projecte.

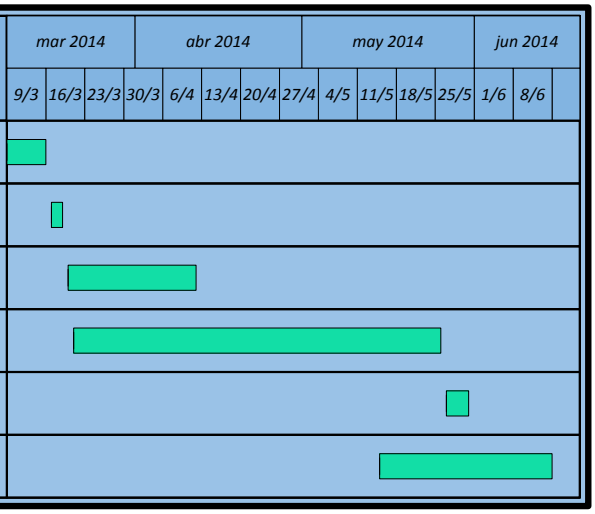
	Item	Cost
Hores de treball	320h	320*10 = 3200€
Equipament informàtic	PC	1000€
	Portàtil	500€
	SmartPhone	450€
Total		5150€

Taules 2. Taula de costos del projecte.

2.4 Diagrama de Gantt

Planificació de les tasques a realitzar en el projecte durant la tasca de planificació.

Id.	*Nom de la tasca*	*Inici*	*Finalització*	*Duració*
1	Especifiació de requeriments	09/03/2014	15/03/2014	7d
2	Planificació projecte	17/03/2014	18/03/2014	2d
3	Disseny	20/03/2014	11/04/2014	23d
4	Codificació	21/03/2014	25/05/2014	66d
5	Validació	27/05/2014	30/05/2014	4d
6	Realització del informe	15/05/2014	14/06/2014	31d



Taules 3. Diagrama de Gantt.

2.5 Podem abordar el projecte?

Analitzant tots els punts anteriors es pot veure que projectes similars més treballats han tirat endavant i no hi ha límits tecnològics que frenin el desenvolupament del projecte.

En el meu cas em crec capaç de tirar endavant aquest projecte sense pecar d'ambiciós per no enganxar-me els dits per falta de temps, i durant el desenvolupament analitzar el progrés per estudiar la inclusió d'alguna funcionalitat més a les inicialment pensades.

3 Descripció de la tecnologia utilitzada

3.1 Plataforma mòbil utilitzada

Per dur a terme aquest projecte he utilitzat la plataforma Android i en concret la API 4.4 kitkat, la qual es la ultima versió disponible.

He decidit utilitzar aquesta plataforma a pesar de que existeixen altres alternatives com iOS, i altres plataformes amb un mercat menor, com Windows Phone i BlackBerry.
Algunes característiques que m'han fet decantar per aquesta plataforma són que te el mercat més gran en aquests moments, més d'un 70% dels SmartPhones que es venen porten el sistema operatiu Android. Al pertànyer a la comunitat de codi obert, no té costos de llicencia, distribució i desenvolupament. En iOS per a poder desenvolupar aplicacions es necessita una llicència que costa 99$ al any [18]. El desenvolupament de les aplicacions és multi plataforma, en canvi iOS només es pot desenvolupar sobre Mac OS X. Utilitza els llenguatges Java i C++ àmpliament utilitzats. També m'ha fet decantar que disposo d'un dispositiu mòbil Android i per tant la possibilitat de utilitzar l'aplicació desenvolupada. Per tant són suficients raons de pes per escollir la plataforma Android.

3.2 Android

3.2.1 Una mica d'història

L'Android va ser iniciat dins d'una empresa, que va ser qui li va donar el nom, Android Inc., fundada el 2003 i el 2005 Google va comprar aquesta empresa.

El novembre del 2007 l'Open Handset Alliance surt a la llum amb l'objectiu de desenvolupar estàndards per a dispositius mòbils. I en aquesta data descobreixen el seu nou producte, l'Android, una plataforma construïda sobre el nucli Linux.

El desembre del 2008 s'afegeixen a l'Open Handset Alliance catorze nous membres, incloent-hi ARM Holdings, Atheros Communications, Asustek Computer Inc., Garmin Ltd., PacketVideo, Softbank, Sony Ericsson, Toshiba Corp. i Vodafone Group Plc.

El maig del 2008 es concedeixen els premis del primer Android developer challenge, que és un concurs per a aconseguir les millors aplicacions per a l'Android. Amb aquest concurs, i les edicions següents, s'aconsegueix una gran repercussió i grans aplicacions.

El setembre del 2008 s'allibera la primera versió de la plataforma, la 1.0, però encara no hi havia cap dispositiu.

Finalment l'octubre del 2008 apareix el primer dispositiu, T-Mobile o HTC Dream, venut inicialment als Estats Units i posteriorment a la resta del món. I el nombre de dispositius no ha deixat de créixer, i ja són centenars de dispositius diferents.

El febrer del 2011 es fa oficial la primera versió orientada no a smartphones sinó a ordinadors de tauleta o tablet PC, amb millores en la interfície, suport de videotrucada i capacitats per a un maquinari molt més potent.

Actualment es parla de més de 100.000 activacions de dispositius diàriament amb la plataforma Android, i de gairebé 200.000 desenvolupadors que ja han creat desenes de milers d'aplicacions, i no paren de créixer.

Aquesta gran quantitat de novetats i la velocitat amb què apareixen provoca que hi hagi moltes versions disponibles a cada moment, i que hem de tenir en compte quan desenvolupem per a Android.

3.2.2 Llicencia Apache en Android

La llicència [16], [17] preferida per al projecte de codi obert d'Android és la Apache Software License , Version 2.0. La major part del programari d'Android té llicència Apache 2.0. Pot haver excepcions que no utilitzin aquesta llicencia, però seran tractades cas per cas. Per exemple , les parts del nucli de Linux estan sota la llicència GPLv2 amb excepcions del sistema, que es poden trobar a kernel.org .

Permet al usuari del software la llibertat de fer-lo servir per a qualsevol propòsit, distribuir-lo, modificar-lo, i distribuir versions modificades d'aquest software.
No exigeix que les versions modificades del software es distribueixin fent servir la mateixa llicencia, ni tan sols que es tinguin de distribuir com a software lliure. La llicencia Apache només exigeix que s'informi als receptor que s'ha utilitzat codi amb llicencia Apache. Per tant els que reben el codi amb aquesta llicencia tenen la llibertat d'utilitzar el codi de la forma que vulguin, incloent el seu us en productes de codi tancat.

3.2.3 Què és?

Android és una solució completa de programari per a dispositius mòbils. Inclou tota una pila d'aplicacions, des del sistema operatiu fins a programari intermedi (middleware) i aplicacions clau. També inclou eines per a desenvolupar a la plataforma, principalment utilitzant el llenguatge de programació Java. Tot això amb llicència de codi lliure Apache, la qual cosa obre moltes possibilitats.

Taules 3. Arquitectura de components Android.

3.2.4 Nucli

El nucli o kernel de l'Android està basat en el nucli Linux, amb algunes modificacions per a adaptar-lo a les necessitats de l'Android.

Aquest serveix com a capa d'abstracció del maquinari a què han d'accedir les aplicacions, de tal manera que si es necessita un element de maquinari (hardware), l'aplicació simplement demanarà el recurs en genèric, sense haver de conèixer el model exacte de maquinari. Per exemple, si volem accedir a la càmera, no importarà el tipus, la definició ni el fabricant; simplement accedirem a la càmera per mitjà del controlador corresponent.

3.2.5 Biblioteques

La capa que se situa just sobre el nucli la componen les biblioteques natives de l'Android. Aquestes biblioteques estan escrites en C o C++ i compilades per a l'arquitectura de maquinari específica del telèfon, tasca que normalment fa el fabricant, que també s'encarrega d'instal·lar-los al terminal abans de posar-lo en venda. L'objectiu que té és proporcionar funcionalitat a les aplicacions, per a tasques que es repeteixen amb freqüència, per a evitar haver de codificar-les cada vegada i garantir que es duen a terme de la manera més eficient.

Aquestes són algunes de les biblioteques que s'inclouen habitualment:

Gestor de superfícies.

Scalable graphics library (SGL).

OpenGL for embedded systems (OpenGL ES).

Biblioteques multimèdia.

WebKit.

Secure sockets layer (SSL).

SQLite.

Biblioteca C de sistema (libc).

3.2.6 Entorn d'execució

El component principal de l'entorn d'execució d'Android és la màquina virtual Dalvik, component que executa totes i cadascuna de les aplicacions no natives de l'Android. Les aplicacions es codifiquen normalment en Java i són compilades, però no per a generar un executable binari compatible amb l'arquitectura de maquinari específica del dispositiu Android. En lloc d'això, es compilen en un format específic per a la màquina virtual Dalvik, que és la que les executa. Això permet compilar una única vegada les aplicacions i distribuir-les ja compilades tenint la total garantia que es podran executar a qualsevol dispositiu Android que disposi de la versió mínima del sistema operatiu que requereixi cada aplicació.

3.2.7 Marc d'aplicacions

La capa següent la formen totes les classes i serveis que utilitzen directament les aplicacions per a fer les seves funcions i que, òbviament, es recolzen en les biblioteques i en l'entorn d'execució que ja hem detallat. La majoria dels components d'aquesta capa són biblioteques Java que accedeixen als recursos per mitjà de la màquina virtual Dalvik.

3.2.8 Aplicacions

La capa superior d'aquesta pila programari la formen, com no podria ser de cap altra manera, les aplicacions. En aquest sac s'inclouen totes les aplicacions del dispositiu, tant les que tenen interfície d'usuari com que no, tant les natives (programades en C++) com les administrades (programades en Java), tant les que vénen de sèrie amb el dispositiu com les instal·lades per l'usuari.

Aquí hi ha també l'aplicació principal del sistema: Inici (Home), també anomenada de vegades llançador (launcher), perquè és la que permet executar altres aplicacions, proporciona la llista d'aplicacions instal·lades i mostra diferents escriptoris on es poden col·locar accessos directes a aplicacions o fins i tot petites aplicacions incrustades o widgets, que són també aplicacions d'aquesta capa.

El principal que cal tenir en compte d'aquesta arquitectura és que totes les aplicacions, tant si són les natives de l'Android com les que proporciona Google, les que inclou de sèrie el fabricant del telèfon o les que instal·la després l'usuari, utilitzen el mateix marc d'aplicació per a accedir als serveis que proporciona el sistema operatiu. Això implica dues coses: que podem crear aplicacions que usin els mateixos recursos que usen les aplicacions natives (res no està reservat o és inaccessible) i que podem reemplaçar qualsevol de les aplicacions del telèfon per una altra que triem.

Aquest és el verdader potencial de l'Android i el que el diferencia de la seva competència: control total per part de l'usuari del programari que s'executa al seu telèfon.

3.3 Eines de desenvolupament

Per la realització de la part de desenvolupament d'aquest projecte he utilitzat les següents eines.

Eclipse és un entorn de desenvolupament integrat (IDE) multi plataforma de codi obert. A traves de varis plugins es pot personalitzar per dur a terme diferents tasques.

Android Development Tools (ADT), és un plugin per Eclipse que proporciona un conjunt d'eines que s'integren amb l'IDE d'Eclipse. Ofereix accés a moltes característiques que l'ajuden a desenvolupar aplicacions d'Android ràpidament. ADT ofereix una eina de disseny d'interficies d'usuari per a la creació ràpida de prototips, dissenys i construcció de la interfície de l'aplicació.

El **SDK d'Android** proporciona les biblioteques API i les eines de desenvolupament necessàries per crear, provar i depurar aplicacions per Android.

Google Plugin for Eclipse, eina per enllaçar la nostra aplicació amb el App Engine de Google.

A part de les anteriors eines ni han d'altres com l'Android Studio. És similar a Eclipse afegint-hi l'ADT, però ja que l'Android Studio encara esta en fase beta l'Eclipse es una IDE molt més estable.

4 Anàlisis de requeriments

4.1 Requeriments funcionals.

Els requeriments funcionals defineixen una funció del software o els seus components.

R.1 – Obtenir dades dels sensors.

R.1.1 – Iniciar la recollida de dades dels sensors.

R.1.2 – Mostrar la informació captada pels sensors.

R.1.3 – Comprovar que el dispositiu te els sensors: Step Detector i GPS.

R.1.4 – Permetre pausar la recollida de dades dels sensors.

R.2 – Tractar les dades obtingudes dels sensors.

R.2.1 – Permetre guardar les dades obtingudes dels sensors.

R.2.2 – Permetre descartar les dades obtingudes dels sensors.

R.3 – Obtenir dades del servidor.

R.3.1 – Enviar una consulta per obtenir les dades guardades.

R.3.2 – Rebre les dades de la resposta del servidor.

R.4 – Eliminar dades del servidor.

R.4.1 – Enviar un identificador al servidor de la dada que es vol eliminar.

R.4.2 – Rebre confirmació de la dada eliminada.

R.5 – Carregar dades a l'aplicació.

R.5.1 – Un cop s'han rebut les dades, tractar-les per obtenir una bona presentació.

R.5.1 – Mostrar les dades obtingudes del servidor.

R.6 – Connexions servidor.

R.6.1 – Establir connexió al servidor.

R.7 – Manipulació de dispositius d'emmagatzemament.

R.7.1 – Crear fitxers.

R.7.2 – Eliminar fitxers.

R.7.3 – Llegir fitxers.

4.2 Requeriments no funcionals.

Un requeriment no funcional és un requisit o propietat que ha de tenir el sistema en realitzar les seves funcions.

1 Producte

1.1 Eficiència

1.1.1 Permet un nombre indefinit d'usuaris simultanis. En el cas de que el servidor no estigui disponible es podrà reintentar l'operació.

1.2 Fiabilitat

1.2.1 El sistema estarà disponible el 100% del temps

1.3 Portabilitat

1.3.1 Desplegable en qualsevol dispositiu mòbil amb sistema operatiu Android 4.4

1.4 Usabilitat

1.4.1 La interfície serà simple i senzilla de manejar i proporcionarà un alt nivell d'interactivitat lliscant les activitats per veure'n d'altres mitjançant un ViewPager.

2 Organització

2.1 Hardware

2.1.1 Es pot accedir des de: Dispositius mòbils (SmartPhones o tablets), tenint en compte les limitacions de tamany i funcionament dels Smartphones es desenvoluparà una versió que resolgui la visualització i accés als diferents documents des de aquests.

2.2 Instal·lació

2.2.1 No caldrà instal·lar cap software addicional per utilitzar l'aplicació. Només s'haurà de donar el permisos necessaris durant la instal·lació per a que l'aplicació funcioni.

2.3 Implementació

2.3.1 Per desenvolupar l'aplicació s'utilitzarà el llenguatge Android de la SDK aplicant les API i eines Android 4.4 (API 19).

3. Externs

3.1 Legislatius

3.1.1 Privacitat

3.1.2.1 Tota la gestió de dades dels usuaris s'ha d'ajustar en tot moment als requisits imposats per la llei orgànica de protecció de dades.

4.3 Protecció de dades

Actualment no hi ha una política de protecció de dades per als usuaris de l'aplicació, només per part de Google existeix una política de protecció de dades per al desenvolupador de l'aplicació.

Google no es fa càrrec de les dades dels usuaris finals d'una aplicació creada per a un desenvolupador que utilitzi els seus serveis.

4.3.1 Condicions de servei de Google Cloud Plataform

[17] *Last modified: Dec 16, 2013*

These Google Cloud Platform Terms of Service apply to all new accounts for any Cloud Platform services created after February 1, 2014 and will apply to all Google App Engine accounts after May 1, 2014 and to all Google Cloud Storage, Google Prediction API, Google BigQuery Service, Google Cloud SQL, Google Compute Engine and Google Cloud Datastore accounts after March 2, 2014.

Google Cloud Platform

LICENSE AGREEMENT

This Google Cloud Platform License Agreement (the "Agreement") is made and entered into by and between Google and the business entity agreeing to these terms ("Customer"). "Google" means either (i) Google Ireland Limited, with offices at Gordon House, Barrow Street, Dublin 4, Ireland, if Customer's billing address is in any country within Europe, the Middle East, or Africa ("EMEA"), (ii) Google Asia Pacific Pte. Ltd., with offices at 8 Marina View Asia Square 1 #30-01 Singapore 018960, if Customer's billing address is in any country within the Asia Pacific region ("APAC"), or (iii) Google Inc., with offices at 1600 Amphitheatre Parkway, Mountain View, California 94043, if Customer's billing address is in any country in the world other than those in EMEA and APAC.

This Agreement is effective as of the date Customer clicks the "I Accept" button below (the "Effective Date"). If you are accepting on behalf of Customer, you represent and warrant that: (i) you have full legal authority to bind Customer to this Agreement; (ii) you have read and

understand this Agreement; and (iii) you agree, on behalf of Customer, to this Agreement. If you do not have the legal authority to bind Customer, please do not click the "I Accept" button below. This Agreement governs Customer's access to and use of the Service. For an offline variant of this Agreement, you may contact Google at http://www.google.com/enterprise/cloud/contact.html for more information.

2. Provision of the Services.

2.2 Facilities and Data Transfer. All facilities used to store and process an Application and Customer Data will adhere to reasonable security standards no less protective than the security standards at facilities where Google processes and stores its own information of a similar type. Google has implemented at least industry standard systems and procedures to ensure the security and confidentiality of an Application and Customer Data, protect against anticipated threats or hazards to the security or integrity of an Application and Customer Data, and protect against unauthorized access to or use of an Application and Customer Data. Google may process and store an Application and Customer Data in the United States or any other country in which Google or its agents maintain facilities. By using the Services, Customer consents to this processing and storage of an Application and Customer Data. The parties agree that Google is merely a data processor.

2.3 Accounts. Customer must have an Account and a Token (if applicable) to use the Services, and is responsible for the information it provides to create the Account, the security of the Token and its passwords for the Account, and for any use of its Account and the Token. If Customer becomes aware of any unauthorized use of its password, its Account or the Token, Customer will notify Google as promptly as possible.

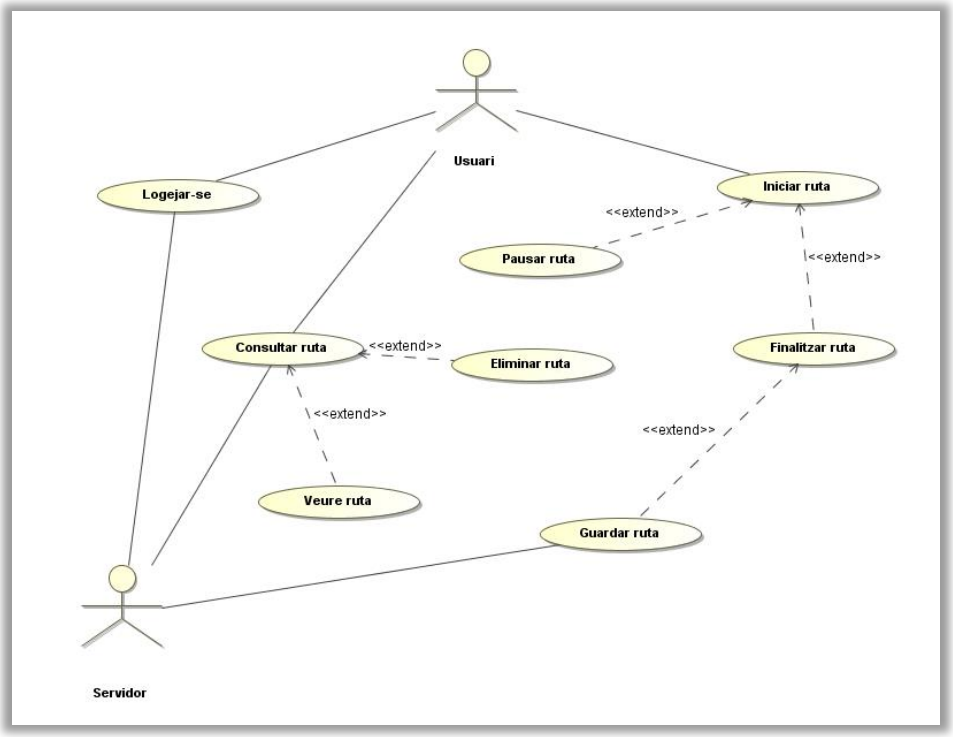
2.4 Privacy Policies. The Services are subject to Google's Privacy Policy. Changes to the Privacy Policy will be made as stated in the applicable policy. In addition, Google is enrolled in the U.S. Department of Commerce Safe Harbor Program and will remain enrolled in this program or another replacement program (or will adopt a compliance solution which achieves compliance with the terms of Article 25 of Directive 95/46/EC) throughout the Term of the Agreement.

4. Customer Obligations.

4.1 Compliance. Customer is solely responsible for its Applications, Projects, and Customer Data and for making sure its Applications, Projects, and Customer Data comply with the Acceptable Use Policy. Google reserves the right to review the Application, Project, and Customer Data to ensure Customer's compliance with the Acceptable Use Policy. Customer is responsible for ensuring all End Users comply with Customer's obligations under the Agreement.

4.2 Privacy. Customer will protect the privacy and legal rights of its End Users under all applicable laws and regulations, which includes a legally adequate privacy notice communicated from Customer. Customer may have the ability to access, monitor, use, or disclose Customer Data submitted by End Users through the Services. Customer will obtain and maintain any required consents from End Users to allow Customer's access, monitoring, use and disclosure of Customer Data. Further, Customer will notify its End Users that any Customer Data provided as part of the Services will be made available to a third party (i.e. Google) as part of Google providing the Services.

4.4 Diagrama de Casos d'Ús



Anàlisis 1. Diagrama de casos d'ús.

4.5 Especificació dels Casos d'Ús

Iniciar Ruta

Especificació d'alt nivell

Cas d'ús: Iniciar Ruta

Actors: Principal: Usuari

Propòsit: Iniciar la recollida de dades dels sensors per a mostrar i emmagatzemar la informació recollida.

Descripció: Un cop l'usuari ha obert l'aplicació i s'ha logejat, podrar iniciar una nova ruta prement el botó Iniciar ruta. Un cop iniciat podrá veure la información recollida a més a més de poder pausar o finalitzar la ruta actual.

Tipus: Primari i essencial.

Especificació expandida

<table>
<tr><td colspan="2">Referències creuades:

<u>Requeriments</u>: R.1.1, R.1.2, R.1.3, R.1.4, R.2.1, R.2.2</td></tr>
<tr><td colspan="2">Curs típic d’esdeveniments:</td></tr>
<tr><td>Accions dels actors</td><td>Respostes del sistema</td></tr>
<tr><td>1. L’usuari ja ha iniciat l’aplicació.</td><td></td></tr>
<tr><td>2. Prem el botó Iniciar ruta.</td><td></td></tr>
<tr><td></td><td>3. Inicia la recollida de dades dels sensors.</td></tr>
<tr><td></td><td>4. Mostra la informació recollida.</td></tr>
<tr><td>5. L’usuari decideix que la ruta s’ha acabat i prem el boto Finalitzar Ruta.</td><td></td></tr>
<tr><td colspan="2">Cursos alternatius:
• La connexió en xarxa falla.
1- Per a guardar una ruta i la informació dels sensors no fa falta un connexió a la xarxa, només per a mostrar el mapa el qual no és prioritari.
• 3. No hi ha tots els sensors necessaris.
1- Mostra quins sensors falten i es continua la recollida d’informació dels sensors que si que estan disponibles.
• 5. L’usuari decideix pausar la ruta.
1- Es pausa la recollida de informació dels sensors i continuarà quant l’usuari decideixi continuar-la.
Quant no hi ha cap sensor disponible es para la tasca de recollida de informació i per tant la ruta.</td></tr>
<tr><td colspan="2">Requeriments especials: Farà falta que la localització GPS estigui activada per una captació de la ruta correcta, i que el dispositiu estigui connectat a la xarxa.</td></tr>
</table>

Consultar ruta

Especificació d'alt nivell

Cas d'ús: Consultar ruta **Actors:** Principal: Usuari, De recolzament: Servidor **Propòsit:** Poder accedir a una ruta guardada prèviament i dur a terme una acció, veure la ruta o eliminar-la. **Descripció:** Si l'usuari ja ha guardat una ruta podrà seleccionar-la i podrà dur a terme dos accions: Veure la ruta, que carregarà totes les dades a l'aplicació, o eliminar la ruta, la qual es borarà del servidor i de la memòria del dispositiu mòbil. **Tipus:** Primari i essencial.

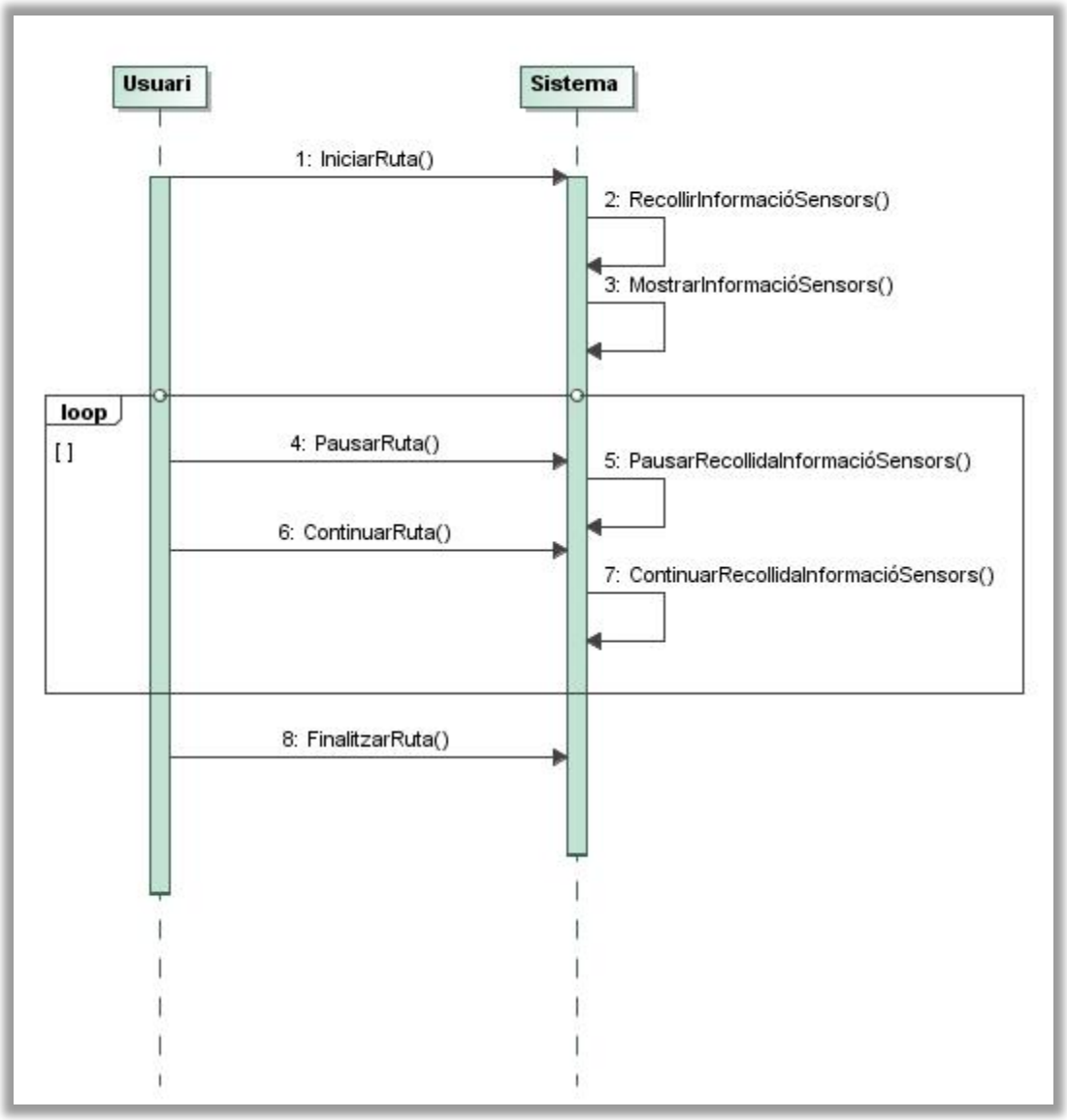
Especificació expandida

<table>
<tr><td colspan="2">Referències creuades:

<u>Requeriments</u>: R.5.1, R.5.2, R.7.1, R.7.2, R.7.3</td></tr>
<tr><td colspan="2">Curs típic d’esdeveniments:</td></tr>
<tr><td>Accions dels actors</td><td>Respostes del sistema</td></tr>
<tr><td>1. L’usuari selecciona una ruta.</td><td></td></tr>
<tr><td></td><td>2. Mostra les opcions que hi ha sobre la ruta.</td></tr>
<tr><td>3. L’usuari selecciona una opció.</td><td></td></tr>
<tr><td></td><td>4. Retorna una resposta depenent de l’opció escollida.</td></tr>
<tr><td colspan="2">Cursos alternatius:
• La connexió en xarxa falla.
1- Es mostrarà un diàleg en el qual es podrà activar la xarxa o sortir.
• 3. Opcions.
1- Carregar ruta.
Es carregarà la ruta escollida guardada prèviament al carregar la llista de les rutes.
Es mostrar les dades de la ruta en l’aplicació.
2- Eliminar ruta.
Es realitza una consulta per eliminar la ruta del servidor.
S’elimina el fitxer de la ruta de la memòria del dispositiu mòbil.
3- Cancel·lar.</td></tr>
<tr><td colspan="2">Requeriments especials: El dispositiu mòbil ha d’estar connectat a la xarxa.</td></tr>
</table>

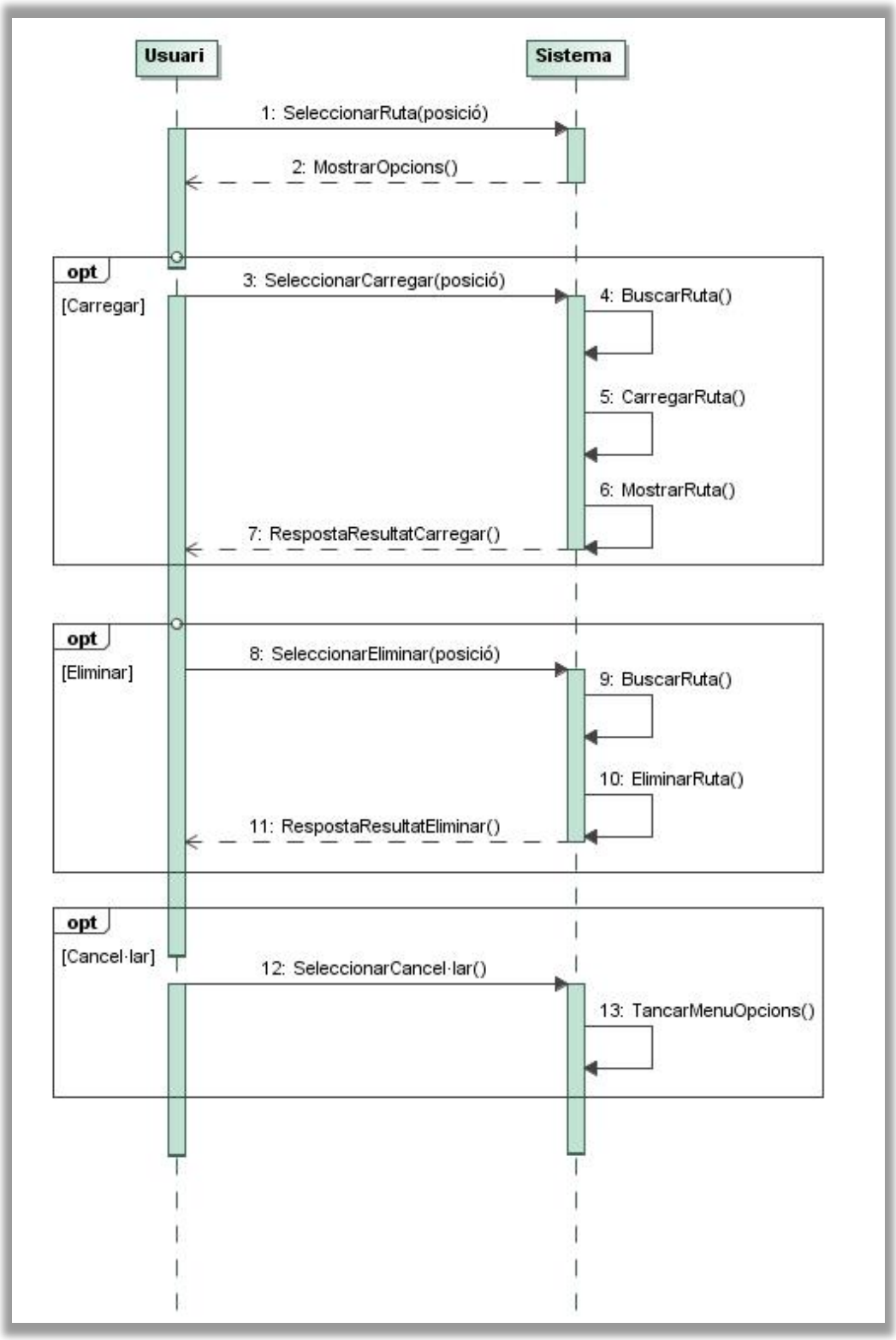
4.6 Diagrames de Seqüència del Sistema

Iniciar Ruta



Anàlisis 2. Diagrama de seqüencia del cas d'us iniciar ruta.

Consultar ruta



Anàlisis 3. Diagrama de seqüencia del cas d'ús consultar ruta.

4.7 Contractes per les operacions

IniciarRuta

- Operació IniciarRuta
    - Nom: IniciarRuta ()
    - Responsabilitats: Iniciar funcions necessàries per a la recollida de informació dels sensors.
    - Tipus: MainFragment.
    - Excepció: No hi ha resposta dels sensors.
    - Precondicions: Identificar-se com a usuari.
    - Postcondicions: Ruta iniciada.
- Operació PausarRuta
    - Nom: PausarRuta().
    - Responsabilitats: pausar la ruta.
    - Tipus: MainFragment.
    - Excepció: No hi ha resposta dels sensors.
    - Precondicions: La ruta esta iniciada.
    - Postcondicions: La ruta esta pausada.

- Operació ContinuarRuta
    - Nom: ContinuarRuta().
    - Responsabilitats: Continuar la ruta.
    - Tipus: MainFragment.
    - Excepció: No hi ha resposta dels sensors.
    - Precondicions: La ruta esta pausada.
    - Postcondicions: La ruta esta iniciada.

- Operació FinalitzarRuta
    - Nom: FinalitzarRuta().
    - Responsabilitats: Guardar la ruta o descartar-la.
    - Tipus: MainFragment.
    - Excepció: No hi ha ruta per a guardar.
    - Sortida: Missatge informant de èxit de la tasca.
    - Precondicions: S'ha realitzat una ruta.
    - Postcondicions: Ruta guardada correctament.

Consultar ruta

- Operació SeleccionarRuta
    - Nom: SeleccionarRuta().
    - Responsabilitat: Escollir una ruta.
    - Tipus: ListFragment.
    - Excepció: No hi ha rutes guardades.
    - Sortida: Identificador de la ruta.
    - Precondicions: Hi ha rutes guardades.
    - Postcondicions: S'ha aconseguir el id de la ruta seleccionada.
- Operació SeleccionarCarregar
    - Nom: SeleccionarCarregar().
    - Responsabilitat: Mostrar la ruta en l'aplicació.
    - Tipus: ListFragment.
    - Excepció: No hi ha connexió a la xarxa.
    - Sortida: Dades de la ruta seleccionada.
    - Precondicions: Es disposa del identificador de la ruta.
    - Postcondicions: Es visualitza la ruta a l'aplicació.

- Operació SeleccionarEliminar
    - Nom: SelecionarEliminar().
    - Responsabilitat: Eliminar una ruta.
    - Tipus: ListFragment.
    - Excepció: No hi ha connexió.
    - Sortida: : Missatge informant de èxit de la tasca.
    - Precondicions: Es disposa del identificador de la ruta.
    - Postcondicions: S'ha eliminat la ruta correctament.
- Operació SeleccionarCancel·lar
    - Nom: SeleccionarCancel·lar().
    - Responsabilitat: Sortir del diàleg de les opcions de la ruta.
    - Tipus: ListFragment.
    - Precondicions: S'està dins el diàleg.
    - Postcondicions: S'ha sortit del diàleg.

5 Disseny de la interfície

5.1 ViewPager

La interfície de l'aplicació ha de mostrar molta informació sobre els sensors, GPS, mapa, i les rutes ja fetes. Per tant necessitava una interfície en la que l'usuari s'hi trobés còmode i la informació es mostrés de forma clara sense la necessitat d'utilitzar menús.

He utilitzat un ViewPager [4], un ViewGroup, el qual és un conjunt de layouts que formen un layout més gran, el qual permet desplaçar-nos entre els diferents layouts o pàgines simplement lliscant les pagines a la dreta o l'esquerra.

D'aquesta manera la interfície de l'aplicació és mes dinàmica i permet tenir sempre a disposició totes les dades a un cop de dit.

El ViewPager està format per una Activity "pare" i diferents Fragment, cada un representa una pàgina. A més a més, per a mostrar en quina pàgina estem a cada moment, i veure si hi ha pàgines adjacents amb aquestes he utilitzat el ViewPagerIndicator [5].

És una llibreria open source creada per Jake Wharton que permet personalitzar un witget per navegar entre les pàgines d'un ViewPager, te la possibilitat d'utilitzar pestanyes i permet la navegació en el ViewPager tant mitjançant el desplaçament gestual com prement les pestanyes.

Captures 3. Captures de les tres pàgines que formen el ViewPager.

Després d'autenticar l'usuari s'inicia la interfície de l'aplicació.

```
//Informacio del usuari
preferences = getSharedPreferences("auth", Context.MODE_PRIVATE);
//ViewPager
mAdapter = new FragmentAdapter(getSupportFragmentManager(),preferences);
mPager = (ViewPager)findViewById(R.id.pager);
mPager.setAdapter(mAdapter);
//Augmentar el limit de pantalles a la memoria
mPager.setOffscreenPageLimit(3);
//Fixar la pantalla per defecte
mPager.setCurrentItem(1);
//ViewPagerIndicator
mIndicator = (TitlePageIndicator)findViewById(R.id.indicator);
mIndicator.setViewPager(mPager);
```

Llistat de codi 1. Inici de la interfície ViewPager.

5.2 Disseny del text

Per a tenir ordenat el format del text en tots els elements de l'aplicació he afegit diferents estils en el fitxer xml styles, i per a seleccionar cada estil he utilitzat l'etiqueta style.

```
<resources>
    <style name="buttons_text" >
            <item name="android:layout_width" >fill_parent</item>
            <item name="android:layout_height" >wrap_content</item>
            <item name="android:textColor" >#FFFFFF</item>
            <item name="android:gravity" >center</item>
            <item name="android:layout_margin" >3dp</item>
            <item name="android:textSize" >30sp</item>
            <item name="android:textStyle" >bold</item>
            <item name="android:shadowColor" >#000000</item>
            <item name="android:shadowDx" >1</item>
            <item name="android:shadowDy" >1</item>
            <item name="android:shadowRadius" >2</item>
        </style>

    <style name="crono" >
            <item name="android:layout_width" >fill_parent</item>
            <item name="android:layout_height" >wrap_content</item>
            <item name="android:textColor" >#FFFFFF</item>
            <item name="android:gravity" >center</item>
            <item name="android:layout_margin" >3dp</item>
            <item name="android:textSize" >25sp</item>
            <item name="android:textStyle" >bold</item>
            <item name="android:shadowColor" >#000000</item>
            <item name="android:shadowDx" >1</item>
            <item name="android:shadowDy" >1</item>
            <item name="android:shadowRadius" >2</item>
        </style>
           .
           .
           .
           .
</resources>
```

Llistat de codi 2. Disseny del text de l'aplicació.

5.3 Disseny dels botons

Tots els botons tenen el mateix format de lletres blanques, fondo negre, costat blanc, i cantonades arrodonides quant no són clicats, i canvia de fondo negre a fondo blau quant es premen.

Per a que tots tinguessin exactament el mateix disseny i es pogués modificar fàcilment si era necessari he utilitzat l'etiqueta android:background en els botons carregant un fitxer xml amb la configuració d'aquests.

Captures 4. Botons del fragment inicial.

Exemple del disseny dels botons, el de la esquerra està premut i el de la dreta no.

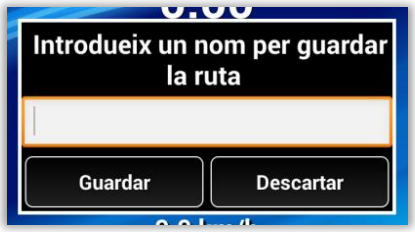
```
<?xml version="1.0" encoding="utf-8" ?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_pressed="true" >
    <shape>
      <solid
          android:color="#0099FF" />
      <stroke
          android:width="1dp"
          android:color="#FFFFFF" />
      <corners
          android:radius="6dp" />
      <padding
          android:left="10dp"
          android:top="10dp"
          android:right="10dp"
          android:bottom="10dp" />
    </shape>
  </item>
  <item>
    <shape>
      <gradient
          android:startColor="#000000"
          android:endColor="#202020"
          android:angle="270" />
      <stroke
          android:width="1dp"
          android:color="#FFFFFF" />
      <corners
          android:radius="6dp" />
      <padding
          android:left="10dp"
          android:top="10dp"
          android:right="10dp"
          android:bottom="10dp" />
    </shape>
  </item>
</selector>
```

Llistat de codi 3. Disseny dels botons de la interfície de l'aplicació.

5.4 Popup window

És una finestra emergent que es pot utilitzar per mostrar una vista d'un layout(fitxer xml on hi ha el disseny de la vista). La finestra emergent és un contenidor flotant que apareix a la part superior de l'activitat actual.

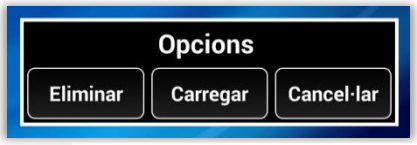
En els popup window mostro diàlegs per a que l'usuari hi interactuï. He utilitzat aquest mètode en comptes de l'alertDialog ja que em permetia controlar millor com l'usuari interactuava amb l'aplicació.



Diàleg que es mostra un cop es vol finalitzar/guardar una ruta.

Captures 5. Diàleg de finalitzar ruta.

Diàleg que es mostra quan hi ha una pulsació de llarga durada sobre una ruta de la llista que es vol consultar.



Com podem veure te un disseny semblant als botons, fondo negre i lletres blanques, amb un costat blanc sense arrodonir les cantonades.

Captures 6. Diàleg de consultar ruta.

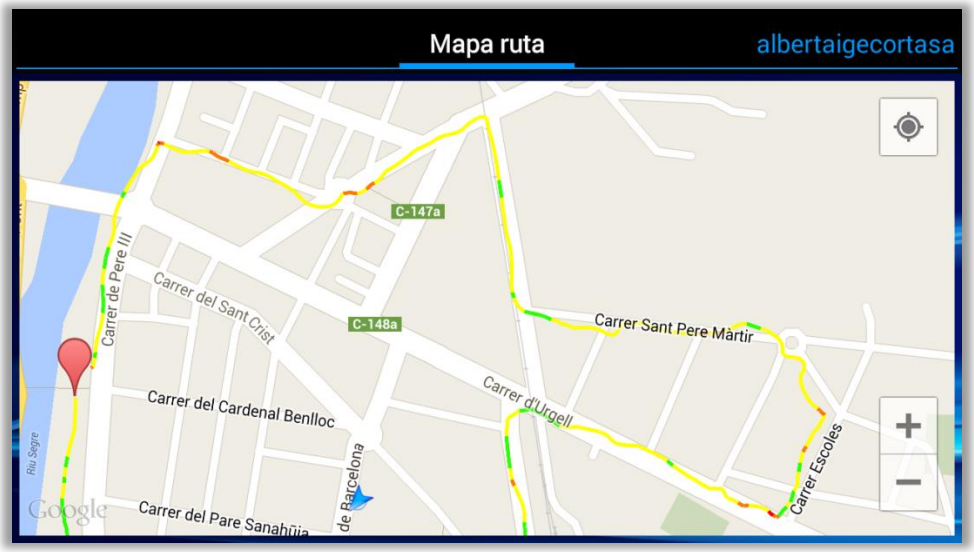
5.5 Icona aplicació

Totes les aplicacions necessiten una icona que les identifiqui, i aquesta no en serà una excepció. Ja que les meves capacitats de disseny no són molt bones he decidit buscar-ne una ja feta. He utilitzat el buscador per a trobar una icona on la principal característica que buscava era que al veure-la es sapigués de que tracta l'aplicació. Finalment vaig trobar aquesta icona amb llicencia gratis per a us personal.

Captures 7. Icona de l'aplicació [16]

5.6 Dibuix del recorregut dins el mapa

Una de les funcionalitats de l'aplicació és observar el ritme, segons els colors en que esta dibuixat cada tram.



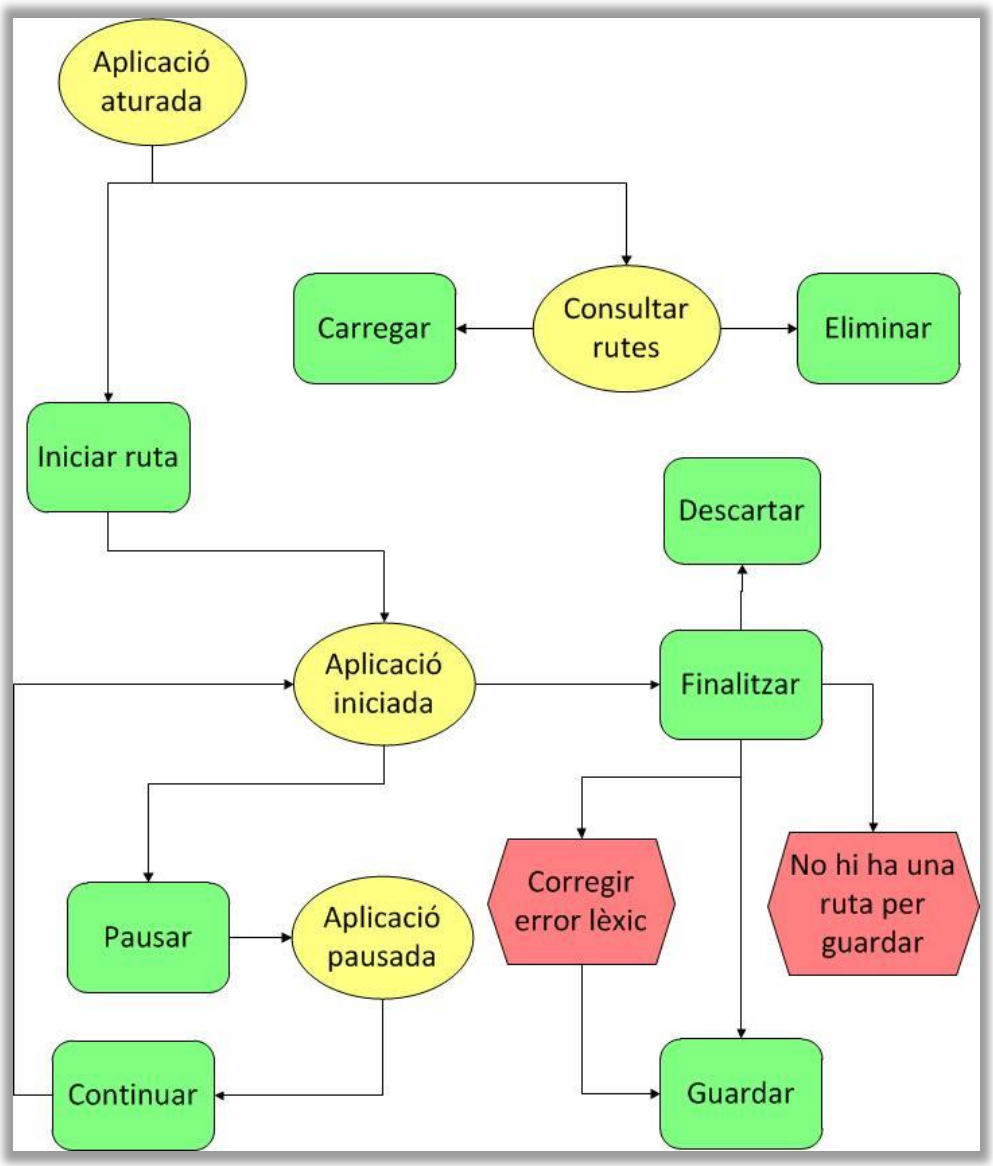
Captura 8. Captura d'una ruta completa realitzada per l'aplicació.

Per aconseguir aquest resultat cada cop que es dibuixa un tram de punt a punt es comprova la velocitat i segons aquesta el color que ha de tenir.

```
//Dibuixar una linea entre dos punts en temps real
public void drawTrack(Location routePoint){
        //Caracteristiques de la linea
        PolylineOptions options = new
        PolylineOptions().width(5).color(tools.getTiming((float)
        (routePoint.getSpeed()*MS_TO_KMH))).geodesic(true);
        //Transformar una Location amb un punt LatLng
        LatLng point = tools.getLatLng(routePoint);
        //Comprovar si hi ha un punt anterior
        if(previousPoint != null){
                //Afegir el punt anterior per dibuixar una linea fins al punt actual
                options.add(previousPoint);
        }else{
                //Dibuixar marcador al inici de la ruta
                map.addMarker(new MarkerOptions().position(tools.getLatLng(routePoint)));
        }
        //Afegir punt
        options.add(point);
        //Dibuixar punt
        map.addPolyline(options);
        //Guardar punt anterior per tenir una referencia d'on començara la seguent linea
        previousPoint = point;
}
//Retorna el color segons la velocitat
public int getTiming(float speed){
        int color = Color.BLACK;
        if(speed>=S_TURTLE)
        {
                color = Color.parseColor(C_RED);
        }
        if(speed>S_SLOW)
        {
                color = Color.parseColor(C_ORANGE);
        }
        if(speed>S_MEDIUM)
        {
                color = Color.parseColor(C_YELLOW);
        }
        if(speed>S_FAST)
        {
                color = Color.parseColor(C_GREEN);
        }
        return color;
}
```

Llistat de codi 4. Mètodes que dibuixen les línies en el mapa.

5.7 Diagrama de transicions entre estats.



Anàlisis 4. Diagrama de transicions entre estats de l'aplicació.

6 Implementació

6.1 Decisions d'implementació.

Tecnologies o eines utilitzades per a realitzar funcions especifiques de l'aplicació. La persistència de les dades, format del fitxer XML per guardar les rutes i utilització de nous sensors.

6.1.1 Backend

L'aplicació necessita guardar les dades en un servidor per a que l'usuari pugui accedir-hi desde qualsevol dispositiu mòbil Android. Per tant queden descartades qualsevol base de dades en local.

A continuació es descriuen dos tecnologies bastant complertes: NodeJs, i Google App Engine, que poden satisfer aquesta necessitat, però tenen grans diferencies.

6.1.1.1 NodeJs

NodeJs [19] és una plataforma construïda sobre JavaScript engine de codi lliure de Google, per construir fàcilment aplicacions ràpides de xarxa escalables. NodeJs utilitza un model d'entrada/sortida dirigida no bloquejant per esdeveniments que ho fa lleuger i eficient, ideal per a aplicacions en temps real de dades que s'executen a través de dispositius distribuïts.

- Avantatges:
  - Flexibilitat en escollir tecnologies.
  - Habilitat de fer canvis i evolucionar.
- Inconvenients:
  - S'ha de construir tots els serveis un mateix.

Exemple de codi utilitzat per a guardar una ruta amb la plataforma NodeJs.

Codi servidor en JavaScript.

Aquesta part de codi en JavaScript en el servidor controla les peticions http, el postsControllerApp executa diferents accions segons la url que el servidor rep per un port específic.

```
//Controlador de les Urls
var postsControllerApp = {
        //Mostrar la llista de rutes en el navegador
        index: function(req, res) {
                res.render("data-list", {posts: Post.getAll()});
        },
        //Veure una ruta a traves del navegador
        view: function(req, res) {
                if (!req.postPlantilla) throw Error("Not found!");
                res.render("data", {post: req.postPlantilla});
        },
        //Guardar una ruta desde el dispositiu mobil
        saveData: function(req, res){
                console.log("Dades guardades");
                var newData = new Post({name: req.body.name, date: req.body.date, distance:
                req.body.distance, avgV: req.body.avgV, maxV: req.body.maxV, passes:
                req.body.passes, time: req.body.time, track: req.body.track, user:
                req.body.user});
                res.redirect('/posts');
                newData.save();
        },
        //Eliminar una ruta desde el navegador
        deletee: function(req, res){
                req.postPlantilla.delete();
                res.redirect('/posts');
        },
        //Controlar el identificador de cada ruta guardada
        param: function(req, res, next, id) {
                req.postPlantilla = Post.find(id);
                next();
        },
        //Enviar ruta
        send: function(req, res){
                console.log("Dades enviades");
                var rute = Post.find(req.postPlantilla.id);
                res.write(rute.name);
                res.end();
        }
};

//Processar el parametre :postid utilitzant el metode creat en el controlador
app.param("postid", postsControllerApp.param);

//Procesamos las rutas.
app.get('/', function(req, res){
   res.render('index');
});

//URLs
//Mostrar rutes web
app.get('/posts', postsControllerApp.index);
//Mostrar dades de la ruta web
app.get('/posts/:postid', postsControllerApp.view);
//Eliminar ruta
app.delete('/posts/:postid', postsControllerApp.deletee);
//Guardar ruta
app.post('/posts/save', postsControllerApp.saveData);
//Obtenir ruta android
app.get('/posts/get/:postid', postsControllerApp.send);
```

Llistat de codi 5. Codi del servidor NodeJs.

Codi client Android en Java.

En la part de client, tan si es vol guardar una ruta al App engine o al NodeJs s'ha de recollir les dades i passar-les per paràmetre a un thread asíncron i executar-lo. Dins d'aquest thread es realitzen les connexions pertinents i les comprovacions posteriors.

```
//Guardar les dades al NodeJs
Button btnGuardarNode = (Button) popupView.findViewById(R.id.bGuadarNode);
btnGuardarNode.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View v) {
                //Crear un objecte json per enviar les dades
                String json = "";
                try {
                        JSONObject jsonObject = new JSONObject();
                        jsonObject.accumulate("name", name.getText());
                        jsonObject.accumulate("date", tools.getDate());
                        jsonObject.accumulate("distance", meters);
                        jsonObject.accumulate("avgV", avg);
                        jsonObject.accumulate("maxV", max);
                        jsonObject.accumulate("passes", stepCount);
                        jsonObject.accumulate("time", Ttime/1000);
                        jsonObject.accumulate("track",
                        gpx.save(segment,name.getText().toString()).getValue());
                        jsonObject.accumulate("user", preferences.getString("user", null));
                        json = jsonObject.toString();
                        se = new StringEntity(json);
                } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                } catch (JSONException e) {
                        e.printStackTrace();
                }
                //Executar AsyncTaks
                new PostData().execute(se);
                popupWindow.dismiss();
        }
});
private class PostData extends AsyncTask<StringEntity, Void, HttpResponse> {
        @Override
        protected HttpResponse doInBackground(StringEntity... params) {
                HttpResponse httpResponse = null;
                HttpClient httpclient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(IP_SERVER+PORT_SERVER+"/posts/save");
                try {
                        httpPost.setEntity((StringEntity)params[0]);
                        //Informar que les dades enviades estan en format json
                        httpPost.setHeader("Accept", "application/json");
                        httpPost.setHeader("Content-type", "application/json");
                        httpResponse = httpclient.execute(httpPost);
                } catch (ClientProtocolException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                return httpResponse;
        }
         @Override
         protected void onPostExecute(HttpResponse result) {
                 //Comprovar si l'accio s'ha completat satisfactoriament
                 if(result!=null){
                        Toast.makeText(v.getContext(), "ha
                        funcionat",Toast.LENGTH_LONG).show();
                         //Recollir les dades guardades
                         new GetData().execute(id);
                         id = id+1;
                 }else{
                         Toast.makeText(v.getContext(), "no ha
                         funcionat",Toast.LENGTH_LONG).show();
                 }
         }
}
```

Llistat de codi 6. Codi del client Android que utilitza el servidor NodeJs.

Degut a la complexitat que suposa crear tota la part de backend del servidor amb aquesta eina he descartat utilitzar-la.

6.1.1.2 Google App engine

Google App Engine [1] és una plataforma com a servei (PAAS) que ofereix crear i executar aplicacions en la infraestructura de Google. Les aplicacions de App Engine són fàcils de construir, fàcil de mantenir i fàcil d'escalar a mesura que el seu trànsit i emmagatzematge de dades van en augment. Amb App Engine, no hi ha servidors que s'hagin de mantenir per un mateix, només s'ha de desplegar l'aplicació i ja està llest per funcionar.

- Avantatges:
  - Fàcil i ràpid de configurar.
  - Gran quantitat de característiques integrades.
  - Autenticació a traves de una conta Google.
- Inconvenients:
  - Tancat a la plataforma de Google, poc marge de personalització i portabilitat.

Finalment per les necessitats de l'aplicació he escollit utilitzar el Google App Engine. L'he escollit perquè no tenia la necessitat de que l'aplicació dugues a terme tasques molt especifiques en l'apartat d'emmagatzematge de dades, tot el contrari, només necessitava guardar una taula amb varis camps i recuperar aquesta informació en moments concrets. A més com ja he dit abans disposa de una eina d'autenticació a traves de contes de Google, lo qual fa molt més fàcil la implementació del maneig d'usuaris. En definitiva un dels objectius d'aquest treball era centrar-me en crear una aplicació en Android, no construir un servidor que em permetés interactuar amb l'aplicació. El qual Google App Engine m'ha permès fer molt més fàcilment.

Dins de Google App Engine he utilitzat l'eina Mobile Backend Starter per a realitzar les connexions amb una base de dades al Cloud.

Mobile Backend Starter és una barreja dels beneficis del BaaS i del PaaS. Els quals combinats permeten crear el backend d'una aplicació en un servidor en un sol clic sense tenir que codificar codi en la part de servidor i que a mes et permet disposar dels avantatges de PaaS, com l'alta escalabilitat, disponibilitat i capacitat d'ampliació.

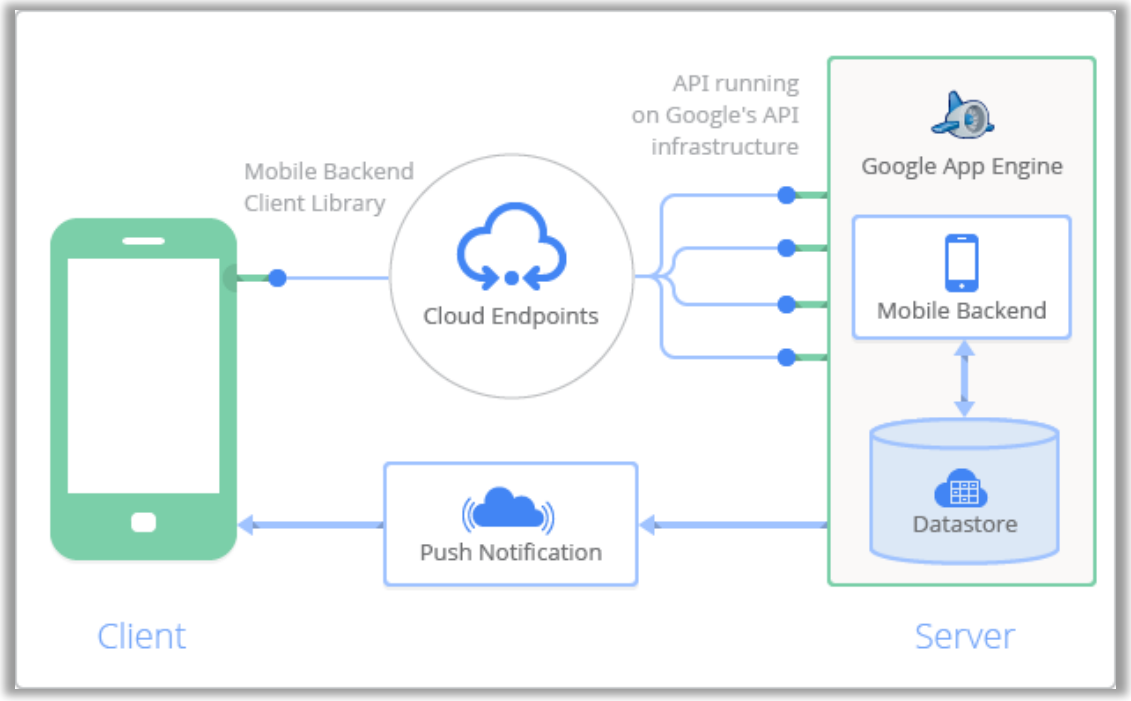
PaaS (Plataform as a Service)

És un model de serveis de cloud computing. En aquest model, l'usuari crea una aplicació o servei utilitzant eines i biblioteques del proveïdor. L'usuari també controla la configuració de la implementació i configuració de programari. El proveïdor proporciona les xarxes, servidors, emmagatzematge i altres serveis que es requereixen per rebre la sol·licitud del usuari.
Facilita el desplegament de les aplicacions o serveis sense el cost i la complexitat de comprar i administrar el maquinari, programari i capacitats d'aprovisionament de hosting.
Es caracteritzen per oferir baix cost inicial, i els costos varien en funció de la utilització dels serveis.

BaaS (Backend as a service)

És un model per proporcionar web i aplicacions mòbils als desenvolupadors, una forma de vincular les seves aplicacions de backend, proporcionant característiques com ara emmagatzematge al núvol, gestió d'usuaris de núvols , les notificacions push, i la integració amb els serveis de xarxes socials.
Tot i que la indústria es relativament recent en la computació en núvol, les tendències indiquen que aquests serveis estan guanyant consumidors empresarials.

Mobile Backend Starter ve amb una sèrie de funcionalitats. Aquestes funcionalitats inclouen una base de dades cloud, Cloud Datastore, autenticació d'usuari, notificacions push, i de publicació/subscripció.



Captura 9. Esquema del funcionament del Mobile Backend Starter.

Per la meva aplicació he utilitzat el Cloud Datastore i l'autenticació d'usuaris.

Cloud Datastore

Com podem veure en la imatge, per a comunicar el client amb el servidor i viceversa s'utilitza una capa intermitja anomenada Cloud Endpoints.
Consisteix en eines, biblioteques i capacitats que li permeten generar APIs i biblioteques de client des d'una aplicació d'App Engine i que es pugen a la part de servidor com a API backend de la nostra aplicació.
Un cop tenim les llibreries a la part de client i l'API a la part del servidor aquests es poden comunicar utilitzant les operacions CRUD.

CRUD, és el acrònim de Create, Read, Update, i Delete. Són les funcions bàsiques en base de dades a la capa de persistència del sotware.

Autenticació d'usuaris.

Permet als usuaris autenticar-se amb una conta de Google el primer cop que s'obra l'aplicació. Implementa OAuth2 per a controlar els tokens d'accés i els d'actualització. Cada cop que s'inicia l'aplicació es demana un token nou.

6.1.2 Format rutes

Un cop l'usuari de l'aplicació finalitza una ruta aquesta s'ha de guardar. Una forma de guardar-la és utilitzant un fitxer de text amb la coordenada i els altres paràmetres necessaris en format llista, però no resulta ser una forma professional en la que guardar una ruta correctament, encara que pugui ser més eficient que altres mètodes com podrem comprovar més endavant.

La forma correcta de guardar aquestes dades és utilitzant estructures de dades estàndard en format XML. M'he centrat en dos dels molts formats que existeixen, GPX i KML, els quals tenen característiques que els diferencien.

6.1.2.1 GPX (GPS eXchange Format)

És un format dissenyat per a l'emmagatzematge i la transferència de dades per a dispositius GPS sense pèrdues. Pot tenir una àmplia gamma de dades molt detallades. El format està disponible al públic a traves de Topographix [11] (http://www.topografix.com/gpx.asp). Té formats explícits segons el tipus de dades, rutes, waypoints i tracks.

- Waypoints <wpt>, són posicions en concret en algun lloc de planeta, es fan servir com a llocs favorits dins d'un sistema GPS portàtil.

- Tracks <trk>, estan fetes d'una successió de waypoints concatenats que dibuixen amb precisió cada racó d'un camí en un mapa. Aquest serà el tipus de dades en el que l'aplicació guardarà el recorregut.

- Rutes <rte>, es composen de la unió entre varis waypoints, poden ser interseccions o ciutats, les quals no són tan precises com les tracks o camins.

Tots aquests tipus de dades tenen uns atributs concrets els quals s'han de respectar per a que la informació que es guardi pugui ser utilitzada en qualsevol dispositiu GPS portàtil.

En el meu cas, per a guardar la ruta he utilitzat el mínim d'atributs possible per a tenir un bon rendiment a l'hora de carregar-ho al mapa. He utilitzat lat(latitud), lon(longitud), i speed(velocitat), els quals són suficients per a que l'aplicació funcioni correctament.

Exemple d'una ruta generada per l'aplicació RunGo en format GPX.

```
<?xml version="1.0" encoding="UTF-8">
<gpx xmlns="http://www.topografix.com/GPX/1/1" creator="RunGo" version="1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX/1/1/gpx.xsd">
<trk>
<name>casa</name>
        <trkseg>
                <trkpt lat="41.7906501" lon="0.8117243">
                <speed>0.0</speed>
                </trkpt>

                <trkpt lat="41.7906297" lon="0.8117438">
                <speed>0.0</speed>
                </trkpt>

                <trkpt lat="41.7905946" lon="0.8117278">
                <speed>0.0</speed>
                </trkpt>

                <trkpt lat="41.7905894" lon="0.8117743">
                <speed>0.0</speed>
                </trkpt>

                <trkpt lat="41.790616" lon="0.8117337">
                <speed>0.0</speed>
                </trkpt>

                <trkpt lat="41.7906228" lon="0.8117079">
                <speed>0.0</speed>
                </trkpt>
        </trkseg>
</trk>
</gpx>
```

Llistat de codi 7. Format GPX d'una ruta.

Captura d'una ruta creada amb el format anterior per l'aplicació.

Captura 10. Captura d'un tram d'una ruta creada per l'aplicació.

6.1.2.2 KML (Keyhole Markup Language)

És un format d'arxiu que s'utilitza per a mostrar dades geogràfiques en un navegador terrestre, com Google Earth, Google Maps i Google Maps per a mòbils. KML [3] utilitza una estructura basada en etiquetes amb atributs i elements afegits. Va ser definit originalment per Keyhole (ara Google) i ara és un estàndard OGC (Open Geospatial Consortium), definició d'estàndards oberts i interpretables dins dels sistemes d'informació geogràfica i de la World Wide Web.

Aquest format especifica una característica, un lloc, una ruta, una imatge o un polígon per a Google Eath.

En la següent captura del Google Eath podem veure a l'esquerra una xinxeta que marca un lloc, i en el centre uns polígons que cobreixen uns edificis de Google.

Captura 11. Captura d'un exemple del format KML.

Exemple d'una ruta en format KML.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"> <Document>
 <name>Rutas</name>
 <Style id="yellowLineGreenPoly">
 <LineStyle>
 <color>7f00ffff</color>
 <width>4</width>
 </LineStyle>
 <PolyStyle>
 <color>7f00ff00</color>
 </PolyStyle>
 </Style> <Placemark>
 <name>nom</name>
 <description>descripcio </description>
 <styleUrl>#yellowLineGreenPoly</styleUrl>
 <LineString>
 <extrude>1</extrude>
 <tessellate>1</tessellate>
 <altitudeMode>absoluto</altitudeMode>
 <coordinates> -112.2550785337791,36.07954952145647,2357
 -112.2549277039738,36.08117083492122,2357
 -112.2552505069063,36.08260761307279,2357
 -112.2564540158376,36.08395660588506,2357
 -112.2580238976449,36.08511401044813,2357
 -112.2595218489022,36.08584355239394,2357
 -112.2608216347552,36.08612634548589,2357
 -112.262073428656,36.08626019085147,2357
 -112.2633204928495,36.08621519860091,2357
 -112.2644963846444,36.08627897945274,2357
 -112.2656969554589,36.08649599090644,2357 </coordinates>
 </LineString> </Placemark>
 </Document> </kml>
```

Llistat de codi 8. Format KML d'una ruta.

Ruta del anterior fitxer KML carregada al Google eath.

Captura 12. Captura de la ruta en format KML.

Finalment he triat el format GPX. Els dos formats permeten un alt grau de personalització, però el que m'ha fet decantar per aquest ha sigut que per convertir un format en l'altre hi ha

algunes restriccions. Hi ha conversors de GPX a KML però no a la inversa i qualsevol dispositiu GPS accepta el format GPX, però no tots suporten el KML.

6.1.2.3 Rendiment en carregar rutes

Al principi he comentat que guardava les rutes en un fitxer de text amb salts de línia per a cada dada. Ho feia per tenir un millor rendiment a l'hora de carregar les rutes, ja que en una sessió llarga poden arribar a ser milers de línies.

Per a mesurar el rendiment he fet dos versions de l'aplicació en les quals una treballa amb fitxers GPX i l'altra en fitxers de text en format llista. En cada versió he implementat un sistema per a calcular el temps en milsegons que tarda en carregar un fitxer.

Per a cada versió he fet 10 captures dels temps i he calculat la mitjana, aquest és el resultat:

- Gpx: 2256,5 ms
- Fitxer en format llista: 823 ms

Com podem veure llegir un fitxer GPX és molt més costos que un fitxer de text amb salts de línia per a cada dada, el qual guarda cada paràmetre en una línia, per tant, molt més ràpid a l'hora de llegir. En canvi per a llegir un fitxer GPX fa falta utilitzar un parser.

6.1.2.4 GPX parser

GPX parser [10] són una sèrie de classes i mètodes que permeten llegir un fitxer GPX i retornar-ne les coordenades i la velocitat de cada punt.

Aquest és el mètode que llegeix un fitxer GPX mitjançant un parser GPX, i dibuixa cada punt del fitxer a un mapa Google Maps. Llegeix tots els punts TrkPt, atribut en el fitxer GPX que conté les coordenades i la velocitat, i les retorna per a dibuixar la línia al mapa.

```
//Llegeix un fitxer Gpx mitjançant un parser Gpx i el dibuixa al mapa
public void readAndDraw(String fileName,GoogleMap map){
        LatLng point=null;
        try {
                //Carregar fitxer GPX
                InputStream is = new FileInputStream(FILE_ROUTE+fileName+FILE_EXTENSSION);
                //Instanciar parser amb el fitxer GPX
                GpxParser gpxP = new GpxParser(is);
                TrkPt trkptLine;
                try {
                        //Obtenir tots els punts del fitxer GPX
                        while((trkptLine = gpxP.nextTrkPt())!=null){
                                point = new LatLng(trkptLine.getLat(), trkptLine.getLon());
                                //Dibuixar linia al mapa
                                tools.drawRoute(point, (float)trkptLine.getSpeed(), map);
                                }
                        CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngZoom(point,
                        ZOOM);
                        map.animateCamera(cameraUpdate);
                } catch (IOException e) {
                        e.printStackTrace();
                }
                tools.previousPoint = null;
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        }
}
```

Llistat de codi 9. Mètode que permet llegir i dibuixar una ruta en format GPX.

Classe principal del parser GPX. Conte el fitxer a parsejar i una funció per a llegir cada punt TrkPt. Cada punt que llegeix el retorna com a una instancia de la classe TrkPt la qual conté les coordenades i la velocitat.

```
//Classe en la qual rep un fitxer Gpx al instanciar-la, i te la funció nextTrkPt la cual
parseja el fitxer i retorna cada punt(<trkpt>) amb les seves dades.
public class GpxParser {
        private InputStream mIs = null;
        private StringBuilder mStringBuilder = new StringBuilder();

        public GpxParser (InputStream is) {
                mIs = is;
        }

        public TrkPt nextTrkPt () throws IOException {
                mStringBuilder.delete(0, mStringBuilder.length());

                int c;
                while ( (c = mIs.read()) != -1 ) {
                        mStringBuilder.append((char)c);

                        TrkPt trkpt = new TrkPt();
                        if (trkpt.parse(mStringBuilder)) {
                                return trkpt;
                        }
                }
                return null;
        }
}
```

Llistat de codi 10. Classe GPXparser.

Aquesta classe és instanciada pel objecte GpxParser per realitzar el pas més important, llegir les dades que seran utilitzades per dibuixar el mapa, i guardar-les en la seva instancia.

```
//Classe que s'encarrega de parsejar cada punt del fitxer Gpx i la qual un cop instanciada te
els parametres Lat, Lon, i Speed necessaris per a carregar la ruta al mapa.
public class TrkPt {
        private final static String START_TAG = "<trkpt ";
        private final static String START_ANY_TAG = "<";
        private final static String END_TAG_EMPTY = "/>";
        private final static String END_TAG_FULL = "</trkpt>";
        private final static String ATTR_LAT = "lat=\"";
        private final static String ATTR_LON = "lon=\"";
        private final static String ELEM_SPEED = "<speed>";
        private double mLat = 0.0;
        private double mLon = 0.0;
        private float mSpeed = (float) 0.0;
        public TrkPt () {
        }
        private boolean parse (String s) {
                int lat = s.indexOf(ATTR_LAT);
                int lon = s.indexOf(ATTR_LON);
                int ele = s.indexOf(ELEM_SPEED);
                if (lat < 0 || lon < 0) {
                        return false;
                }
                int endLat = s.indexOf("\"", lat + ATTR_LAT.length());
                int endLon = s.indexOf("\"", lon + ATTR_LON.length());
                mLat = Double.parseDouble(s.substring(lat+ATTR_LAT.length(), endLat));
                mLon = Double.parseDouble(s.substring(lon+ATTR_LON.length(), endLon));
                if (ele > 0) {
                        mSpeed = Float.parseFloat(s.substring(ele + ELEM_SPEED.length(),
                        s.indexOf("<", ele + ELEM_SPEED.length())));
                }
                return true;
        }
        public boolean parse (StringBuilder s) {
                int startTag = s.indexOf(START_TAG);
                if (startTag < 0) return false;
                int nextTag = s.indexOf(START_ANY_TAG, startTag);
                int endTagEmpty = s.indexOf(END_TAG_EMPTY, startTag);
                int endTagFull = s.indexOf(END_TAG_FULL, startTag);
                if (endTagEmpty + END_TAG_EMPTY.length() == s.length()) {
                        return parse(s.substring(startTag, endTagEmpty +
                        END_TAG_EMPTY.length()));
                } else if (nextTag > 0 && endTagEmpty > 0 && nextTag < endTagEmpty) {
                        if (nextTag == endTagFull) {
                                return parse(s.substring(startTag, endTagFull +
                                END_TAG_FULL.length()));
                        } else if (endTagFull > 0) {
                                return parse(s.substring(startTag, endTagFull +
                                END_TAG_FULL.length()));
                        } else {
                                return false;
                        }
                } else if (nextTag > 0 && endTagFull > 0) {
                        return parse(s.substring(startTag, endTagFull +
                        END_TAG_FULL.length()));
                } else {
                        return false;
                }
        }
        public double getLat() {
                return mLat;
        }
        public double getLon() {
                return mLon;
        }
        public double getSpeed () {
                return mSpeed;
        }
}
```

Llistat de codi 11. Classe TrkPt.

6.1.2.5 Rendiment en guardar rutes

En el moment de guardar una ruta s'ha de crear el text que més endavant es guardarà en un fitxer. Aquest procés pot arribar a ser massa llarg, en una ruta de 30 km pot arribar a tardar entre 3-4 minuts en guardar la ruta.
Per a reduir aquest temps a pocs segons vaig decidir crear el text de la ruta durant la realització de l'activitat física. D'aquesta forma he sigut capaç d'enviar instantàniament la ruta al servidor i l'única demora que hi continua havent és el temps de connexió al servidor i la pujada de la ruta.

6.1.3 Utilització de la memòria física del dispositiu mòbil

L'aplicació utilitza la memòria interna o externa (targetes de memòria), per emmagatzemar el recorregut de la ruta en format GPX.

Per utilitzar-la és necessari afegir el següent permís al manifest de l'aplicació:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Llistat de codi 12. Permís del manifest de l'aplicació.

Al iniciar l'aplicació s'executa la següent funció la qual comprova si el directori existeix i en el cas de que no existeixi es crea en el directori arrel de la memòria.

```
//Crear direcotri per a guardar dades de l'aplicacio
private void createDirecotry(){
        File d = new File(FILE_ROUTE);
        //Si el directori no existeix es crea sino no es fa res
        if(d.isDirectory()==false)
        {
                d.mkdirs();
        }
}
```

Llistat de codi 13. Mètode per a crear el directori.

6.1.4 Step counter i step detector

Una de les funcionalitats noves que incorpora la nova versió d'Android 4.4 són els sensors step counter i step detector. Són un nou hardware de baix consum que detecta les passes quant l'usuari esta caminant, corrent, o pujant escales.

Hi han diferències entre els dos sensors. L'Step counter envia un event amb el total de passes fetes desde l'últim reinici del dispositiu cada cop que es detecta una passa. En canvi l'Step detector envia un event amb valor 1.0 cada cop que es detecta un passa.

En la meva aplicació utilitzo el sensor Step detector, que hem permet contar les passes en un determinat espai de temps en el qual l'usuari ha iniciat una ruta a traves de l'aplicació.

Codi necessari per controlar el sensor Step detector i mostrar-ne les dades [9].

```
//Instanciar SensorManager per a controlar els sensors
private void setSensors(){
        sensorManager = (SensorManager)
        getActivity().getSystemService(Context.SENSOR_SERVICE);
}
//Registrar sensors i comprovar que existeixen
@Override
public void onResume() {
        super.onResume();
        //Step Detector
        stepDetector = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);
        if (stepDetector != null) {
                sensorManager.registerListener((SensorEventListener) this, stepDetector,
                SensorManager.SENSOR_DELAY_NORMAL);
        } else {
                stepCountV.setVisibility(View.GONE);
                if(alredyStarted==false){
                        Toast.makeText(v.getContext(), "Aquest dispositiu no disposa del
                        sensor Step Detector",Toast.LENGTH_LONG).show();
                        alredyStarted=true;
                }
        }
        //GPS
        if(mLocClient.isConnected()){
                mLocClient.requestLocationUpdates(mLocRequest,listener);
        }else{
                mLocClient.connect();
        }
}
@Override
public void onSensorChanged(SensorEvent event) {
//Els canvis només tindran efecte si l'aplicació esta iniciada
        if(started){
                //Quant hi hagui un event del sensor Step detector es mostrara
                if (event.sensor.getType() == Sensor.TYPE_STEP_DETECTOR) {
                        stepCount = stepCount + (int)event.values[0];
                        stepCountV.setText(stepCount+" Passos");
                }
        }
}
```

Llistat de codi 14. Mètodes que controlen el sensor Step Detector.

6.2 Desenvolupament de l'aplicació

L'aplicació està fortament lligada al suport de Google, concretament al App engine, Google Maps i a l'autenticació a traves de contes de Google. Tots aquests serveis s'han de configurar prèviament o durant el desenvolupament a traves de la Consola de APIs de Google.

6.2.1 Crear Endpoints i desplegar

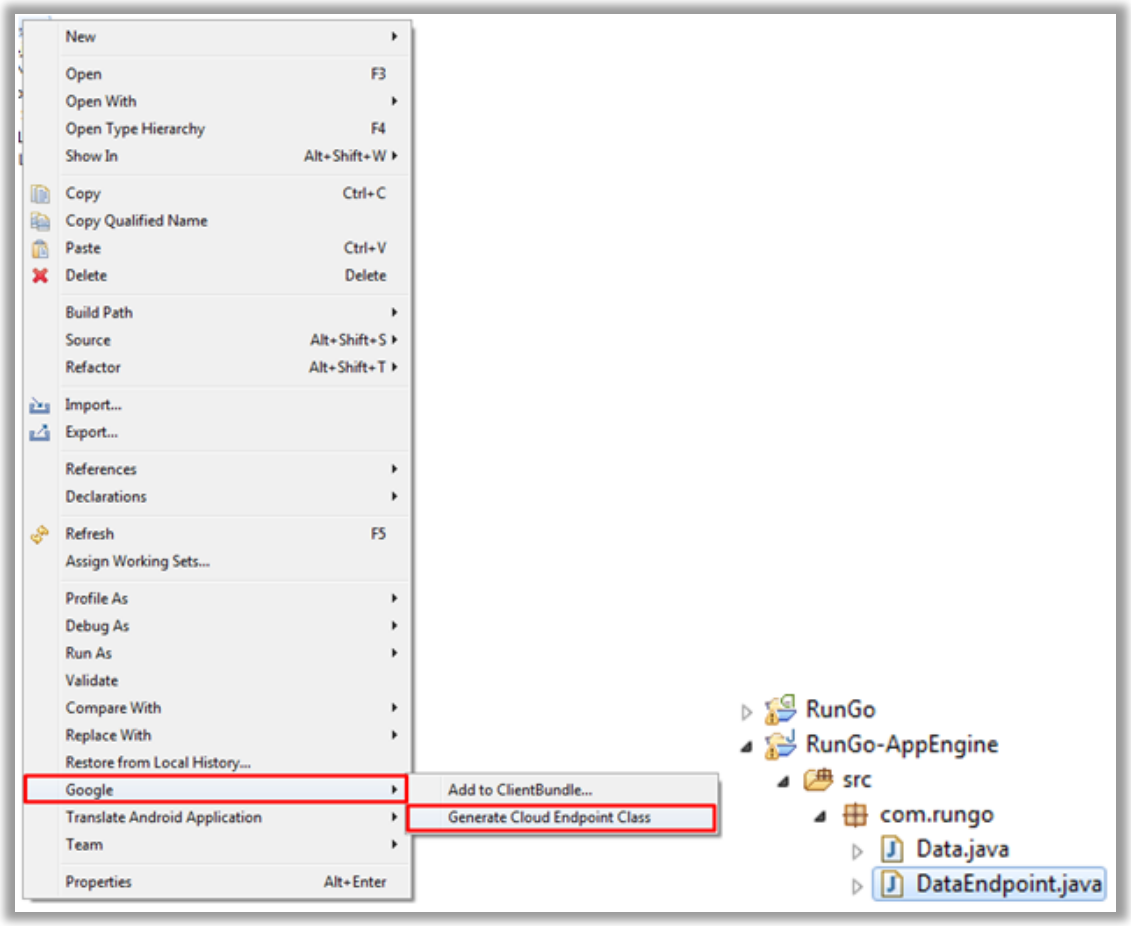
Havent configurat l'App engine anteriorment ja podem començar a crear els Endpoints per l'aplicació i desplegar-los, són un servei que proporciona mètodes remots accessibles pels clients.

Per crear-ne un per l'aplicació primer creem una Entitat la qual defineix la estructura de les dades, la clau primària (@Id), que es generi automàticament (@GeneratedValue(strategy = GenerationType.*IDENTITY*)), i altres atributs.

```
@Entity
public class Data{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Key key;
        private String name;
        private String user;
        private Long time;
        private float maxV;
        private float avgV;
        private Text track;
        private float distance;
        private String date;
        private int passes;

        public Key getKey() {
                return key;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getUser() {
                return user;
        }

        public void setUser(String user) {
                this.user = user;
        }
        .
        .
        .
}
```

Llistat de codi 15. Classe del Endpoint Data.

Després de crear-la generarem el Endpoint d'aquesta Entitat. Clic amb el boto dret sobre la Entitat i en el desplegable Google -> Generate Cloud Endpoint Class.



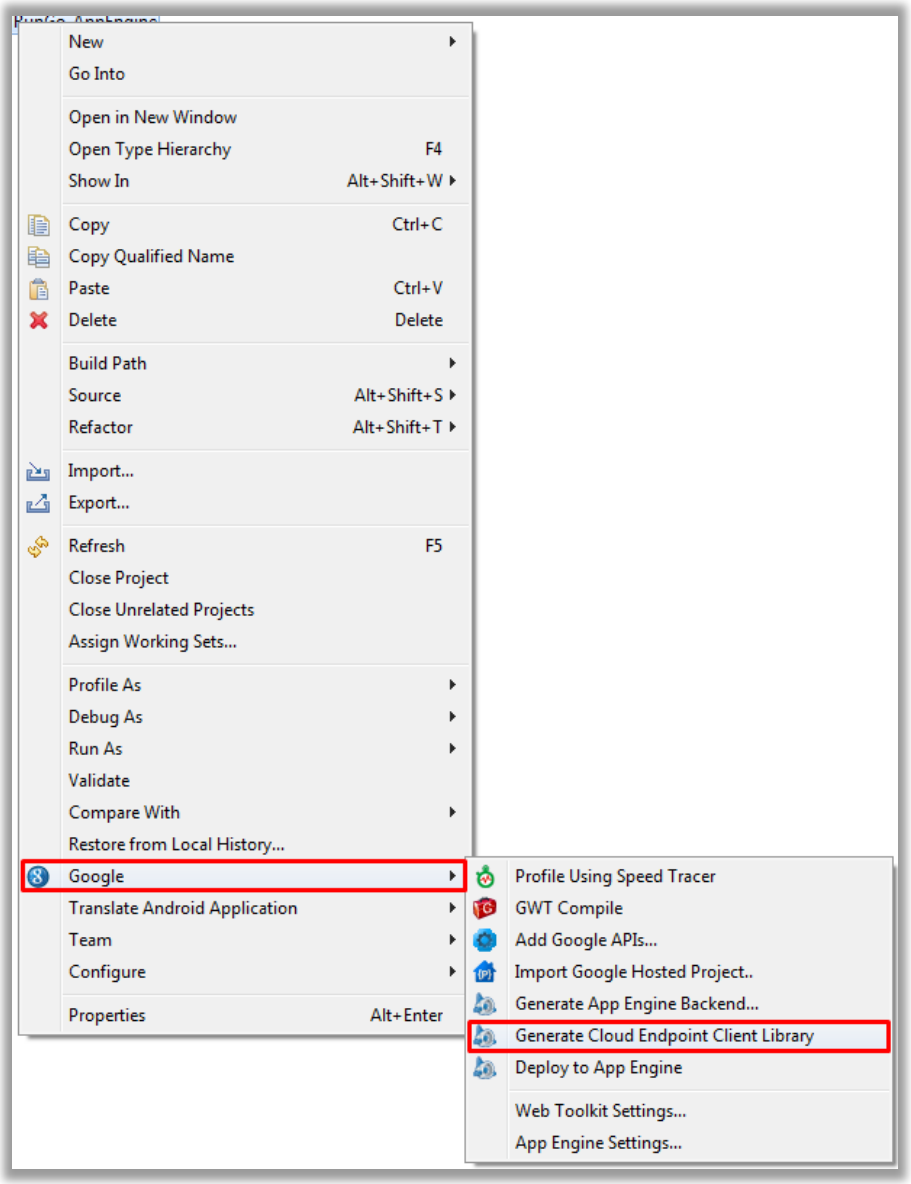
Captura 13. Captura Ecplipse per a generar un Endpoint.

Un cap generada la classe DataEndpoint amb els seus mètodes CRUD s'hi ha de fer una modificació [8], ja que en crear una nova Entitat amb les dades desde el dispositiu mòbil, abans de guardar-la comprova si la Entitat ja existeix. Per fer-ho busca en el Cloud Datastore, però al guardar per primera vegada la Entitat no trobarà la clau perquè encara no s'ha generat. Per tant sempre abans de buscar una clau si aquesta retorna null vol dir que encara no està creada i no existeix en el Cloud Datastore.

```
private boolean containsData(Data data) {
        EntityManager mgr = getEntityManager();
        boolean contains = true;
        try {
                //Inici codi afegit
                if(data.getKey() == null){
                        return false;
                }
                //Fi
                Data item = mgr.find(Data.class, data.getKey());
                if (item == null) {
                        contains = false;
                }
        } finally {
                mgr.close();
        }
        return contains;
}
```

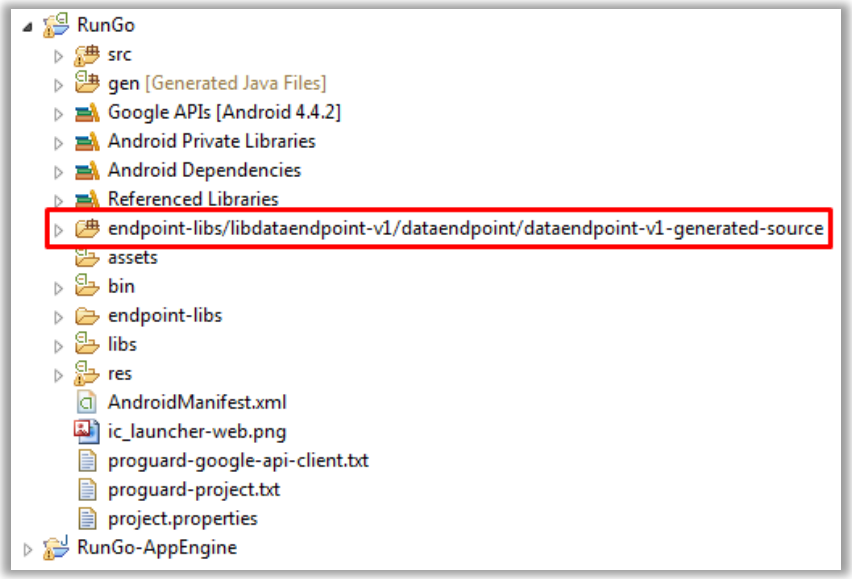
Llistat de codi 16. Mètode del Endpoint modificat.

Després de solucionar aquest problema ja es pot generar les llibreries del Endpoint per a l'aplicació client. En el projecte backend premem el boto dret sobre ell, i al desplegable Google -> Generate Cloud Endpoint Client Library.



Captura 14. Captura Eclipse per generar les llibreries client del Endpoint.

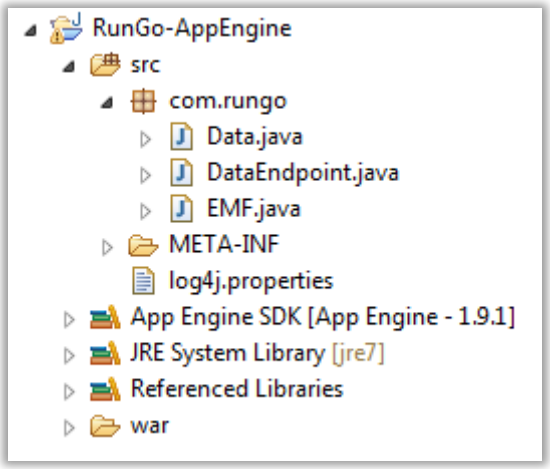
Ara ja es pot utilitzar aquestes llibreries per codificar els mètodes per a guardar, llegir, eliminar i actualitzar dades desde l'aplicació al Cloud Datastore del App engine.



Captura 15. Captura Eclipse dels directoris del projecte de l'aplicació.

Per utilitzar l'aplicació desde el dispositiu mòbil, fan falta unes petites configuracions en el projecte client i s'ha de desplegar el projecte backend al Backend del App engine.

En el projecte veurem una classe anomenada CloudEndpointsUtils, que serveix per a configurar paràmetres de l'aplicació client. S'ha de modificar la següent línia de codi, i canviar el true per false per a executar l'aplicació en un dispositiu mòbil.

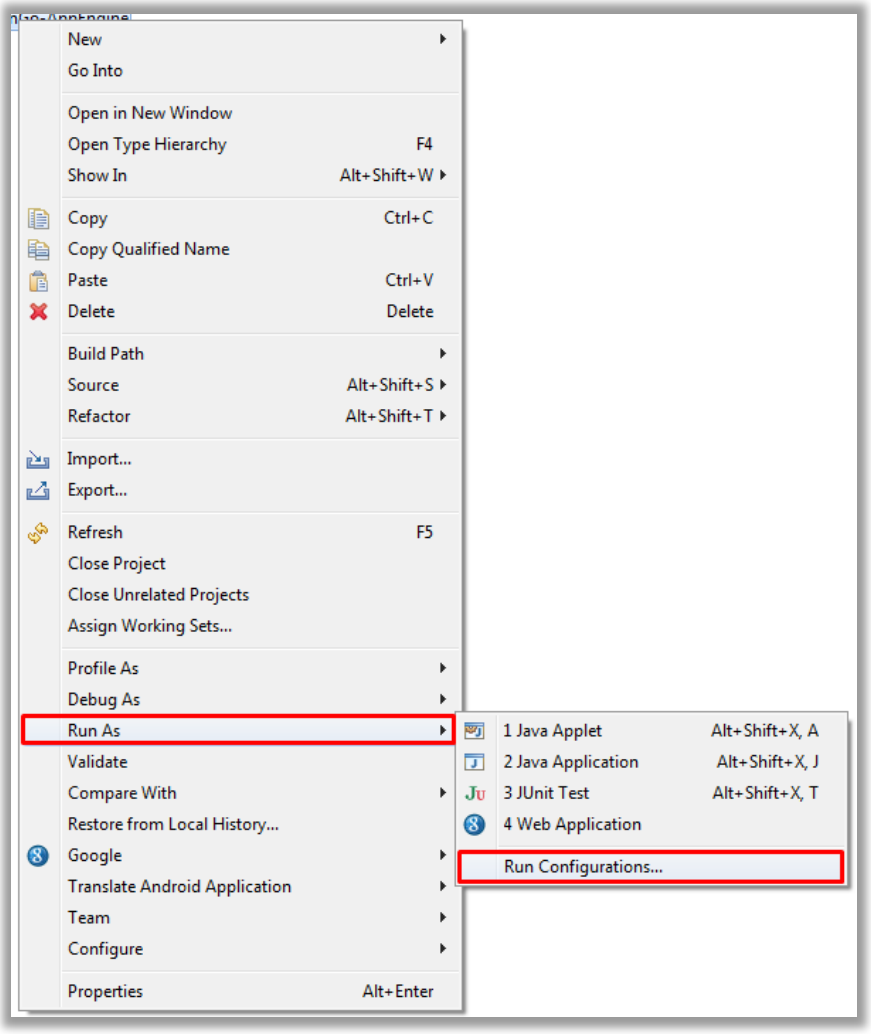


Captura 16. Captura Eclipse dels directoris del projecte App Engine de l'aplicació.

```
/*
* TODO: Need to change this to 'true' if you're running your backend locally using
* the DevAppServer. See
* http://developers.google.com/eclipse/docs/cloud_endpoints for more
* information.
*/
protected static final boolean LOCAL_ANDROID_RUN = false;
```

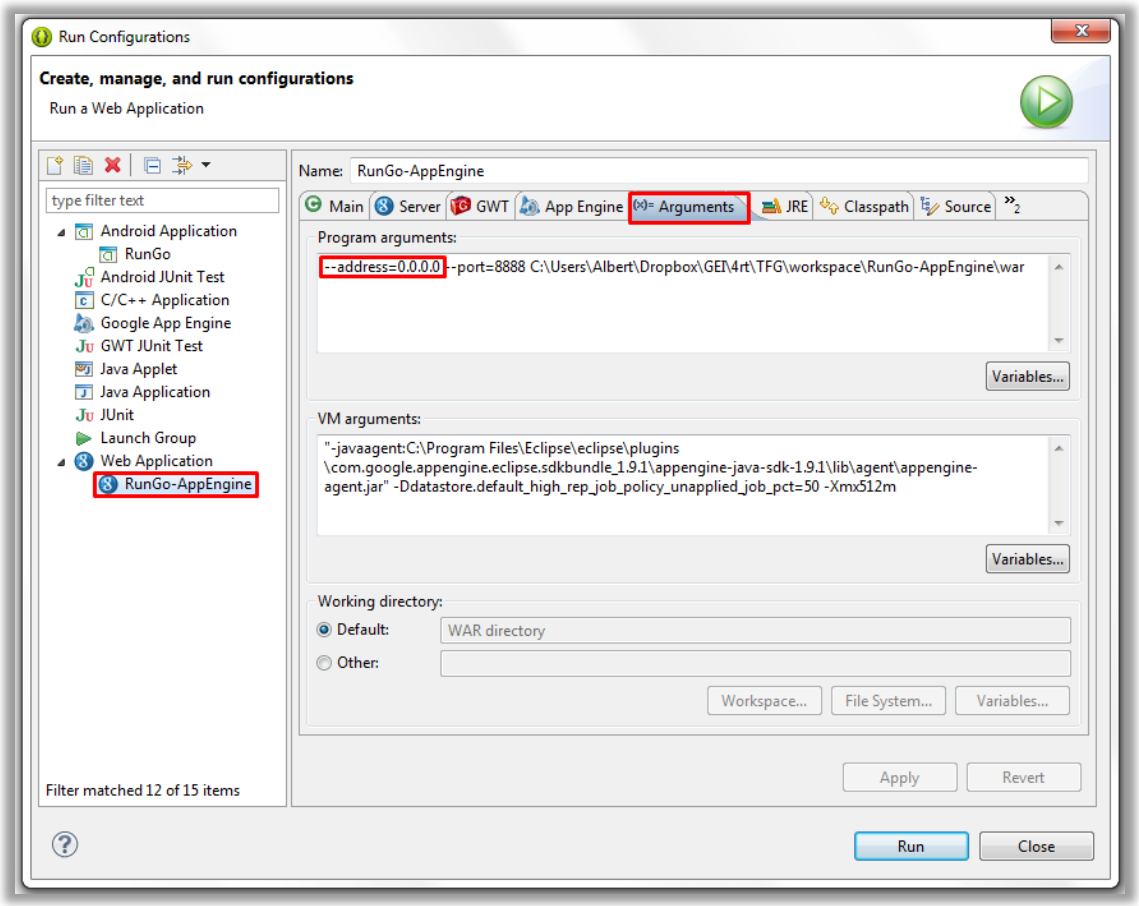
Llistat de codi 17. Variable estàtica per controlar el dispositiu d'execució.

Canviar les configuracions d'execució sobre el projecte backend, boto dret -> Run As -> Run Configurations.



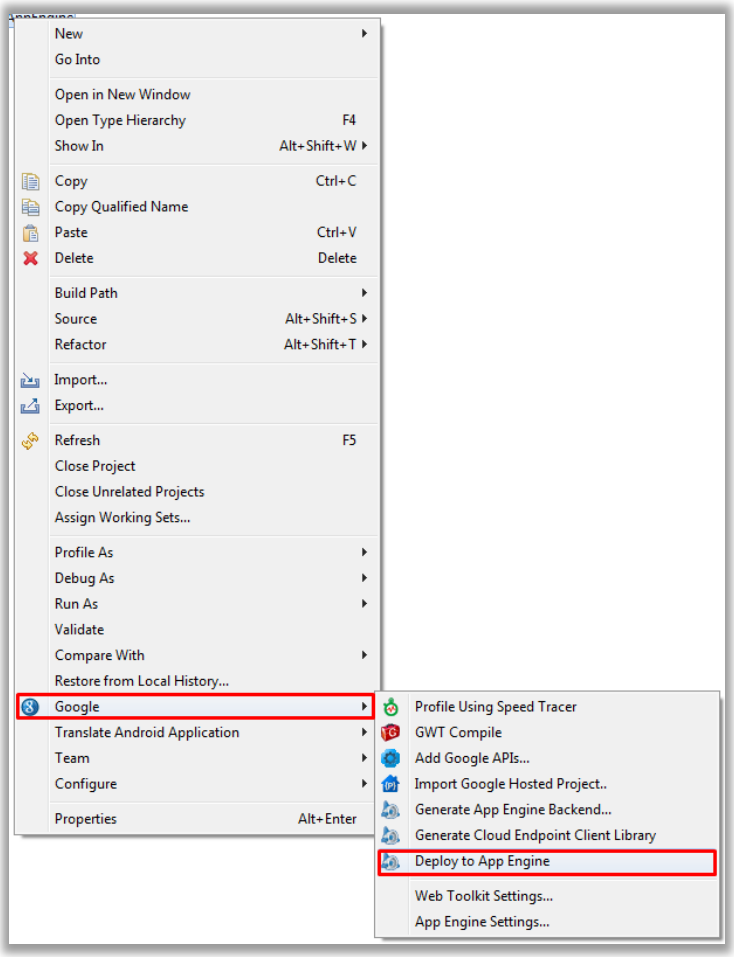
Captura 17. Captura Eclipse per modificar la configuració del App Engine.

En la finestra que s'obrirà seleccionem en el menú de l'esquerra en el nom del projecte backend -> Arguments, i davant de --port=8888 afegim --address=0.0.0.0 i apliquem la configuració.

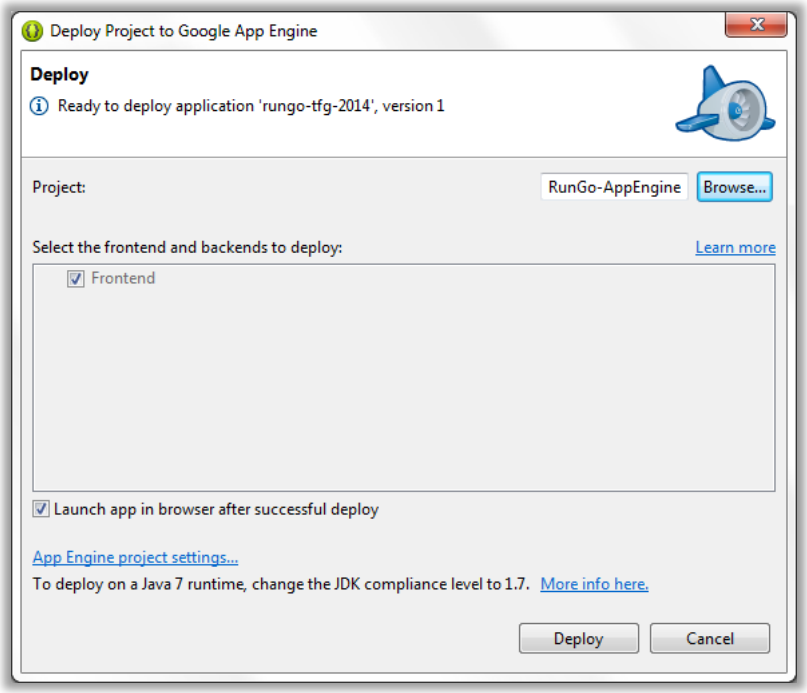


Captura 18. Captura Eclipse de la configuració del App Engine.

Ara ja es pot desplegar el projecte backend en el backend de l'App engine. Sobre aquest projecte, boto dret -> Google -> Deploy to App Engine.

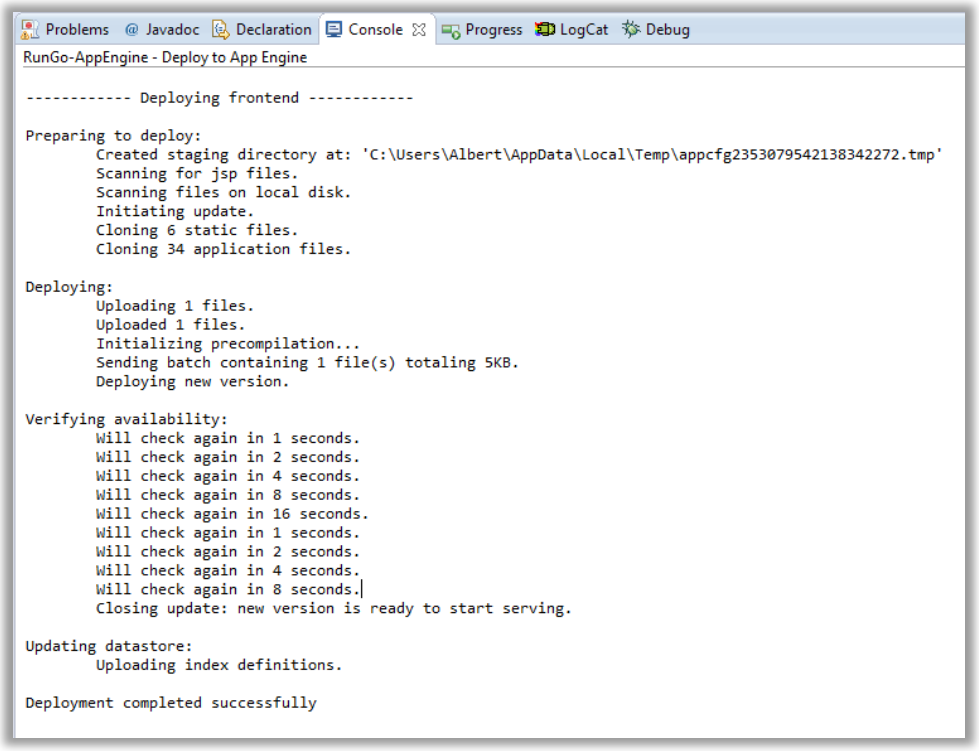


Captura 19. Captura Eclipse per desplegar el projecte de backend.



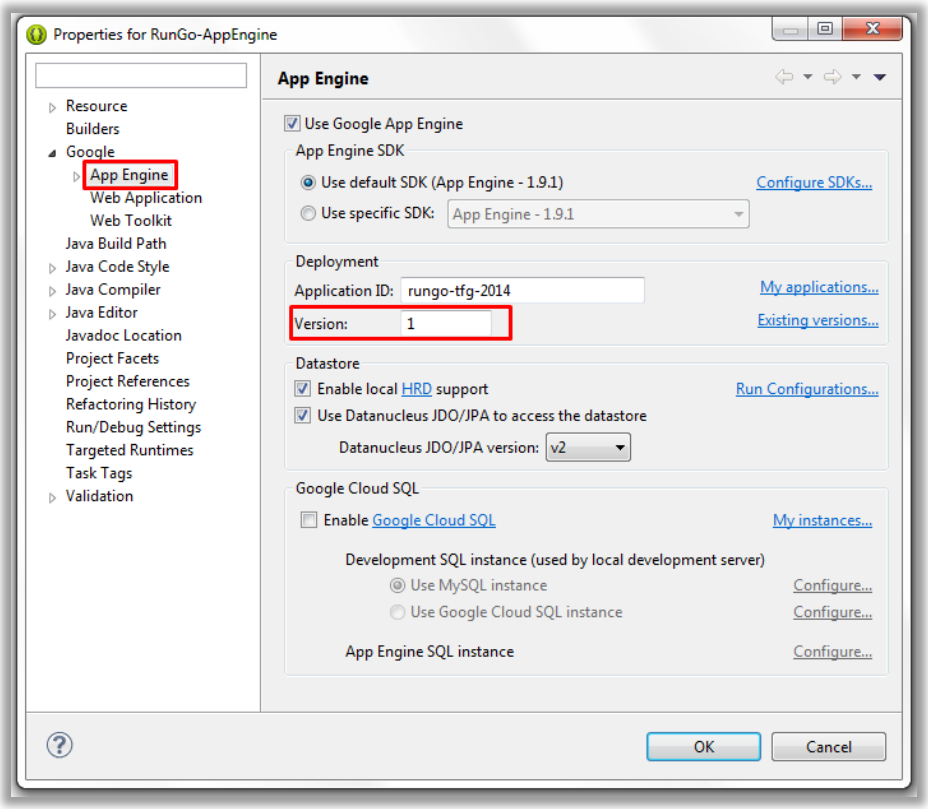
Captura 20. Captura Eclipse de les opcions de desplegar.

En la consola podrem veure el progrés del desplegament i ens informarà quant s'hagi desplegat amb un avis.



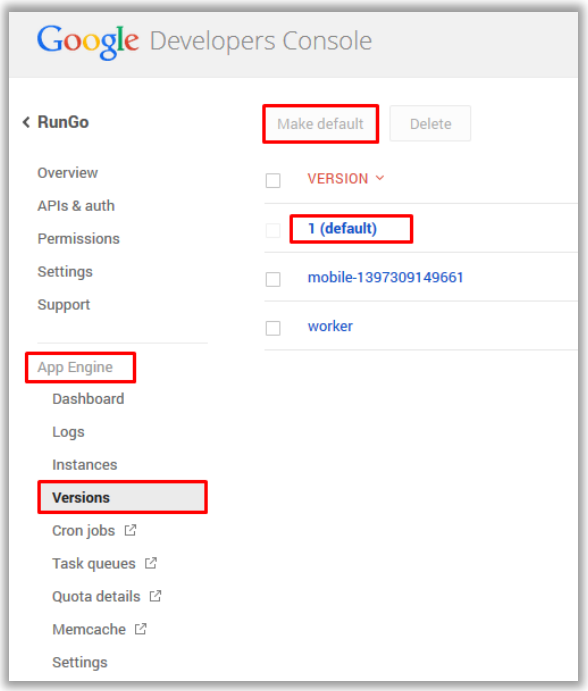
Captura 21. Captura Eclipse de la confirmació del desplegament.

Ara en la Consola de Google s'ha de seleccionar la versió del projecte que em desplegat la qual per defecte és "1", però que es pot canviar en les propietats del projecte backend.



Captura 22. Captura Eclipse de les propietats del App Engine.

Un cop identificada la versió, en l'apartat App Engine -> Versions, hi ha els desplegaments, i les seves versions. Seleccionem la versió "1" i premem el boto Make defaut per utilitzar-la per defecte.



Captura 23. Captura Consola de Google de les versions del desplegades.

Finalment l'aplicació client i el backend que he desplegat ja es poden comunicar per realitzar les accions CRUD.

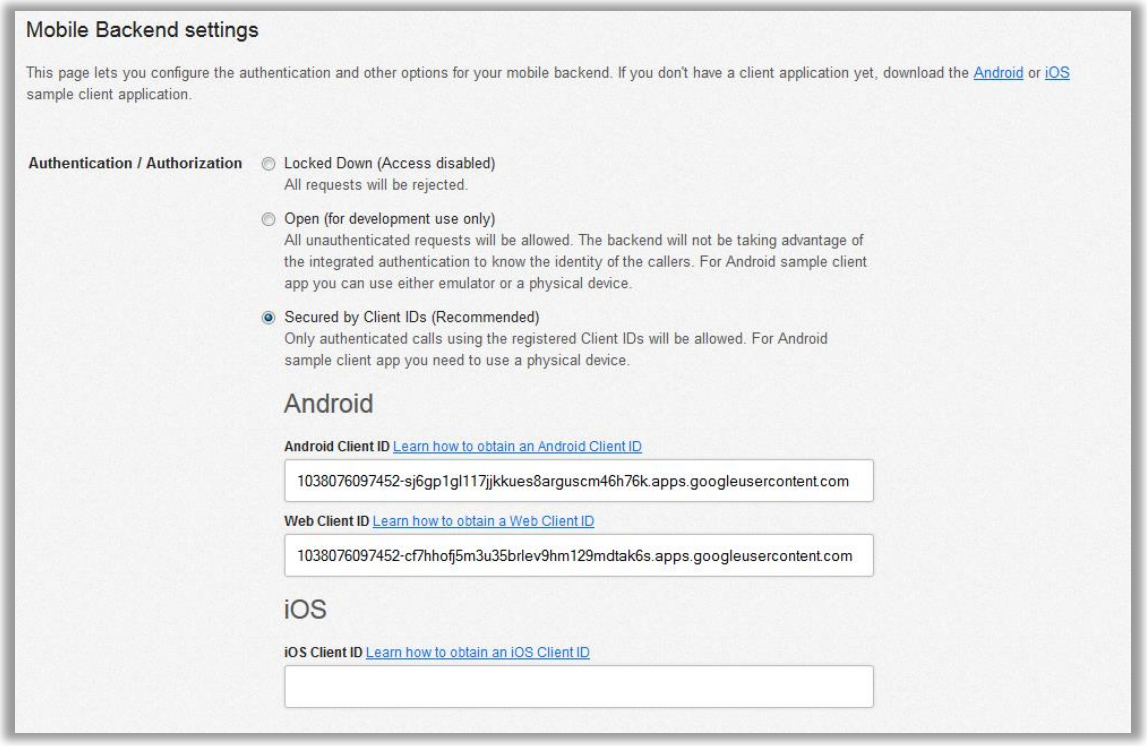
6.2.2 Autenticació

Aprofitant les eines que Google ens proporciona amb el App engine , aprofito per autenticar els usuaris a traves de les seves contes ja existents en els seus dispositius mòbils.

Per activar l'autenticació a traves de contes Google en el manifest del projecte client s'hi ha d'afegir dos permisos i haurem de tornar a la web de Mobile Backend Setting, aquest cop seleccionem Secured by Client IDs i apareixeran tres diàlegs per omplir, només necessitarem els de Android, accedim a la Consola Google per crear-los.

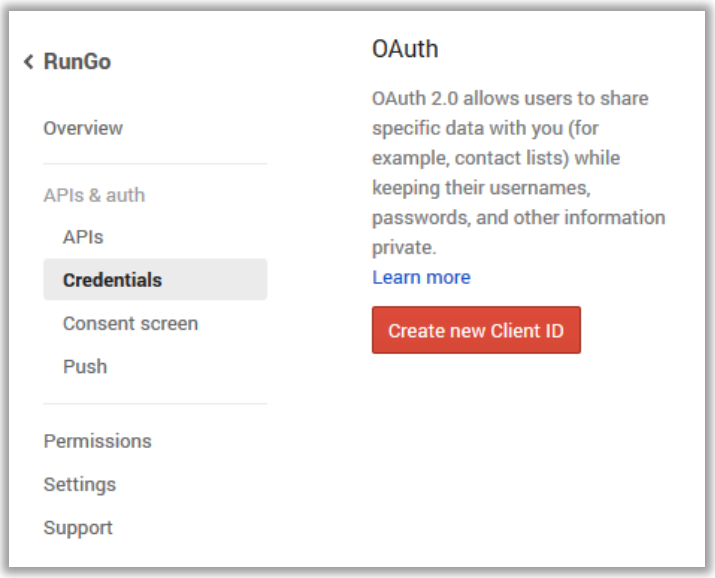
```
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
```

Llistat de codi 18. Permisos del manifest de l'aplicació.



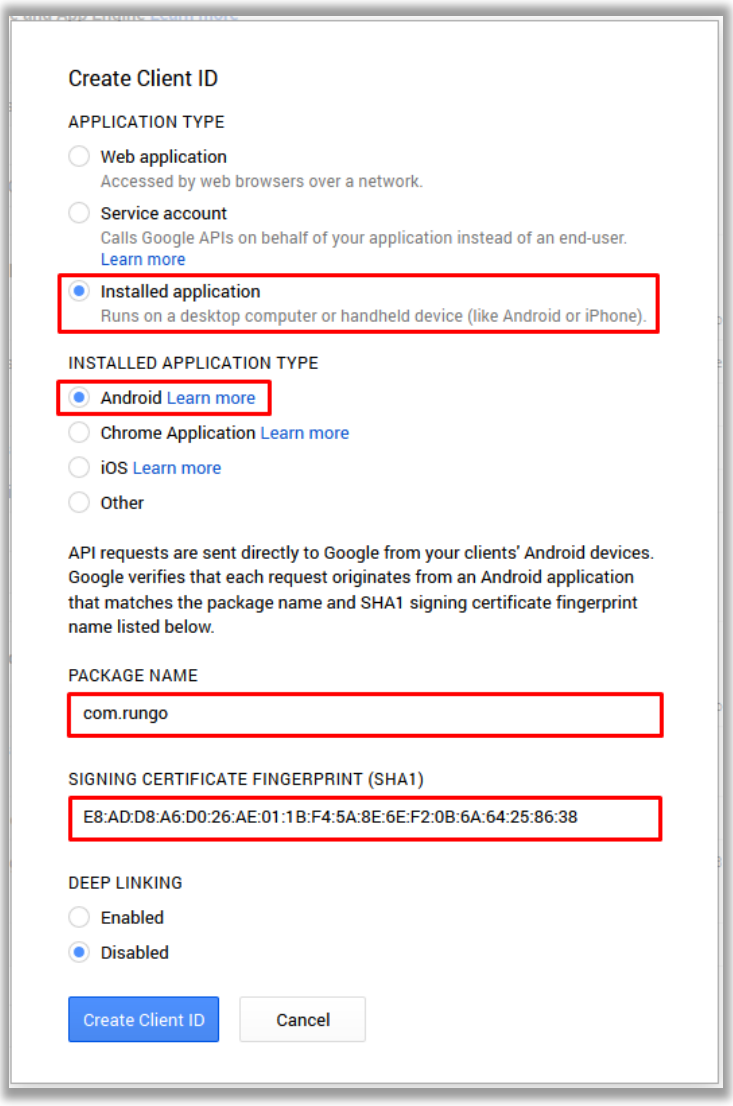
Captura 24. Captura de la configuració del Mobile Backend.

Ens dirigim a APIs & auth -> Credentials -> Create new Client ID.



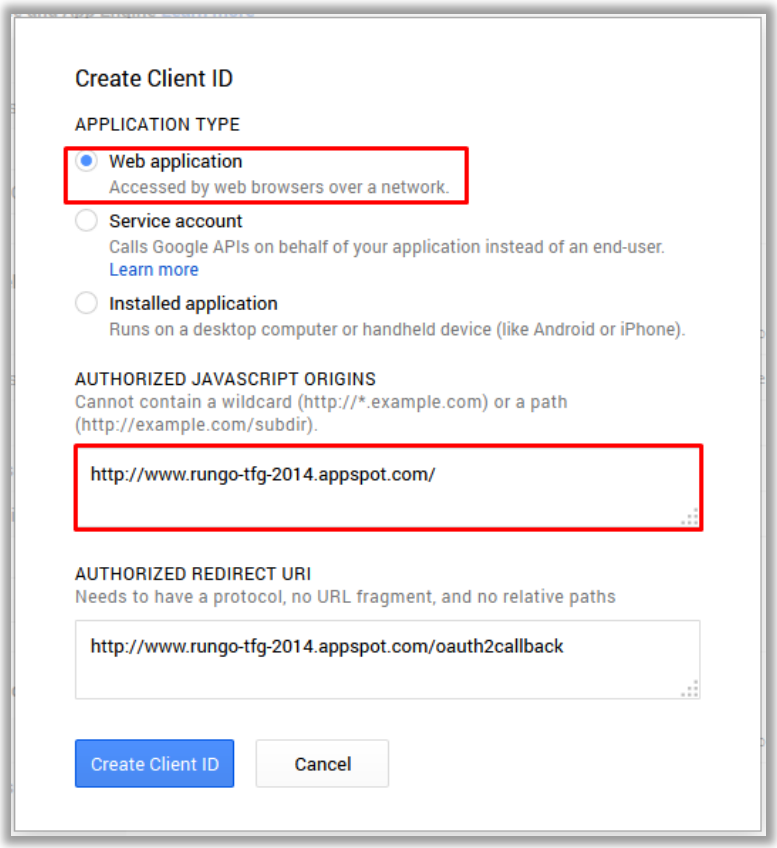
Captura 25. Captura Consola de Google de les credencials d'autenticació.

Per l'Android Client ID seleccionem Installed application, Android. Farà falta el Fingerprint SH1 utilitzat anteriorment, i el nom del paquet de l'aplicació client.



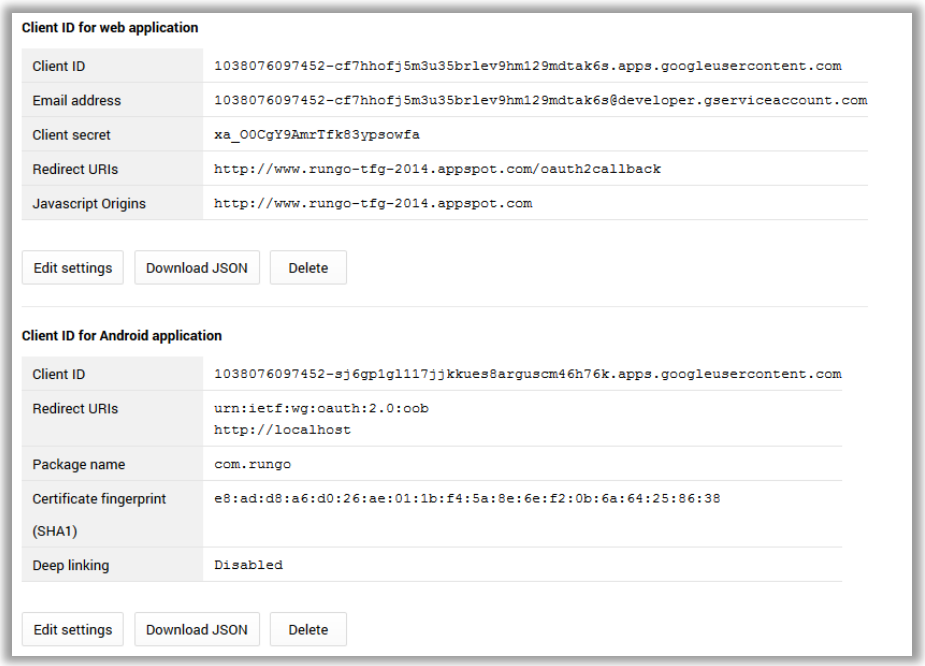
Captura 26. Captura Consola Google de la creació del Client ID.

En el Web Client ID seleccionem Web application, només necessitem la següent url amb el nostre project-id, http://www.PROJECT-ID.appspot.com/, el següent diàleg s'omplirà automàticament al posar la url.



Captura 27. Captura Consola Google de la creació del Web Client ID.

Un cop creats els ids els copiem en els seus corresponents diàlegs del Mobile Backend Settings. Ara ja esta preparat per iniciar la autenticació d'usuaris amb les seves contes de Google.



Captura 28. Captura Consola Google de les keys Client ID.

Un cop configurades les key, en iniciar-se la aplicació es comprova que hi hagi connexió a la xarxa per a poder autenticar al usuari correctament, sinó hi és, es demanarà al usuari que l'activi o surti de l'aplicació [12].
Si hi ha connexió, es continua amb l'autenticació del usuari. Si ja ha estat autenticat anteriorment i la sessió encara no ha caducat s'inicia la aplicació, sinó es realitza un procés d'autenticació.

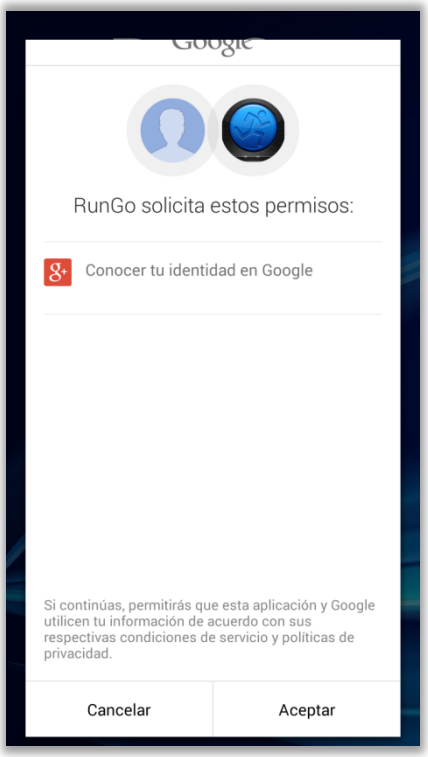
```
//Comprovar que hi ha connexió a la xarxa abans d'autenticar un usuari
if(checkNetwork()
        //Si l'usuari ja ha estat autenticat anteriorment s'iniciara l'aplicació, en cas
        contrari s'haura d'autenticar
        if (authPreferences.getUser() != null && authPreferences.getToken() != null) {
                //Iniciar aplicació
                startApp();
        } else {
                //Autenticar usuari
                chooseAccount();
        }
}
```

Llistat de codi 19. Mètode encarregat de comprovar l'autenticació de l'usuari.

Per iniciar l'autenticació s'inicia un intent, una instancia d'una Activity, per buscar les contes de Google disponibles al dispositiu mòbil.

```
//Iniciar autenticació usuari
private void chooseAccount() {
        Intent intent = AccountManager.newChooseAccountIntent(null, null,new String[] {
        "com.google" }, false, null, null, null, null);
        startActivityForResult(intent, ACCOUNT_CODE);
}
```

Llistat de codi 20. Mètode que inicia el intent de l'autenticació.



Captura 29. Captura intent d'autenticació.

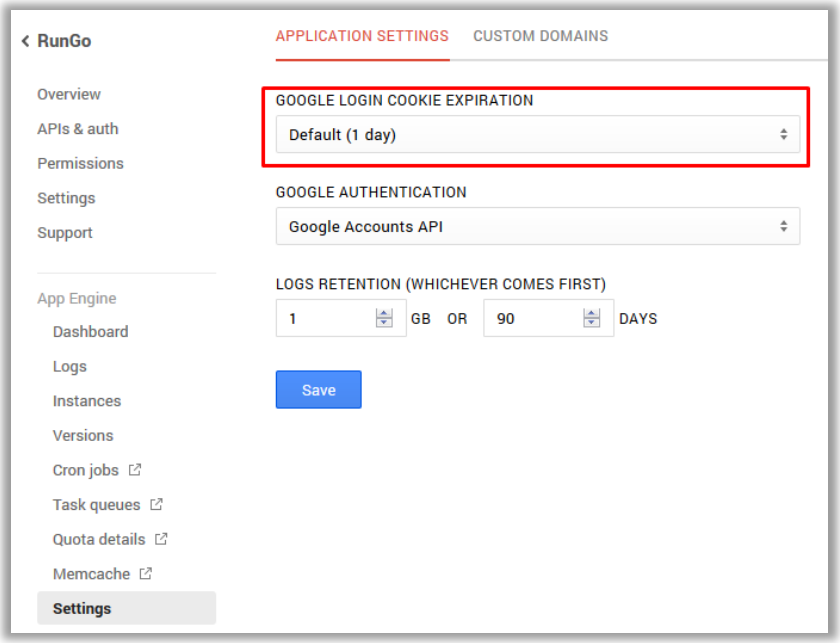
Al acceptar la sol·licitud dels permisos que demana l'aplicació, aquesta funció comprova el resultat.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        //Comprovar el resultat del intent anterior
        if (resultCode == RESULT_OK) {
                if (requestCode == AUTHORIZATION_CODE) {
                        //Agafar el token anteriorment creat
                        requestToken();
                } else if (requestCode == ACCOUNT_CODE) {
                        //Demanar un nou token
                        String accountName =
                        data.getStringExtra(AccountManager.KEY_ACCOUNT_NAME);
                        authPreferences.setUser(accountName);
                        invalidateToken();
                        requestToken();
                }
        }else{
                //L'usuari no ha acceptat les condicions
                Log.e("AuthenticationFailure",String.valueOf(resultCode));
                finish();
        }
}
```

Llistat de codi 21. Mètode que rep el resultat del intent de l'autenticació.

Si l'usuari ha acceptat la sol·licitud de permisos es realitzarà una altra sèrie de comprovacions. Si ja s'ha guardat una cookie de sessió en el anterior login que no estigui caducada procedirà a agafar el token i continuar amb l'inici de l'aplicació. En cas contrari si no existeix la cookie o ja ha expirat invalidarà el token actual i en demanarà un de nou.

El temps d'expiració de les cookies es pot modificar en la Consola Google en App Engine -> Settings.



Captura 30. Captura Consola Google de la configuració de les cookies.

6.2.3 Captura de la informació

Quant l'aplicació esta iniciada els sensors estan actius, però les seves dades no són tractades fins que s'inicia la captura de la ruta amb el boto startPause (Iniciar Ruta), el qual també permet pausar la captura de les dades. El boto stop (Finalitzar), s'encarregarà d'iniciar un diàleg per a guardar les dades de la ruta.

```
//Obtenir dades del usuari
preferences = v.getContext().getSharedPreferences("auth", Context.MODE_PRIVATE);
//Iniciar sensors i GPS
setSensors();
setUpGpsLocation();
//Control boto Iniciar-continuar
startPause.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
                if(started == false){
                        //App Iniciada
                        chronometer.setBase(SystemClock.elapsedRealtime()+timeWhenStopped);
                        chronometer.start();
                        started = true;
                        stop.setEnabled(started);
                        startPause.setText("Pausar");
                }else{
                        //App Pausada
                        timeWhenStopped = chronometer.getBase() -
                        SystemClock.elapsedRealtime();
                        chronometer.stop();
                        started = false;
                        startPause.setText("Continuar");
                }
        }
});
//Control boto Finalitzar
stop.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
                time = getOnSavedTimeCrono();
                chronometer.stop();
                started = false;
                //Mostrar dialeg per a guardar la ruta
                saveDataDialog();
        }
});
```

Llistat de codi 22. Control dels botons Iniciar Ruta i Finalitzar.

Les següents funcions [7] s'encarreguen de mostrar, i guardar les dades en objectes temporals fins que l'usuari no decideix finalitzar la ruta. Per controlar si aquestes funcions han de capturar les dades he creat un boolea (started) el qual si l'usuari ha iniciat l'aplicació es posa en true o en false si l'usuari l'ha pausat o finalitzat.

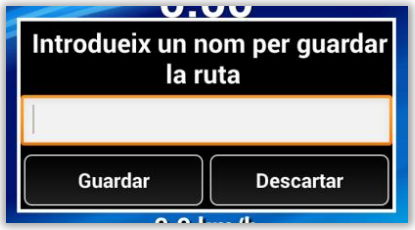
```
//GPS
private void updateLocation(){
        listener = new LocationListener() {
                MapFragment mapFragment = (MapFragment) getFragment(1);
                @Override
                public void onLocationChanged(Location location) {
                        mapFragment.setCameraZoom(location);
                        if(started == true){
                                setFirstLocation(location);
                                //Actualitzar texview distancia
                                setDistanceView(location);
                                //Actualitzar textview velocitat
                                setSpeedParamsView((location.getSpeed()));
                                //Actualizar mapa
                                mapFragment.setLocationMap(location);
                                //Guardar ruta
                                segment = gpx.createSegment(location, segment);
                        }
                }
        };
        mLocClient.requestLocationUpdates(mLocRequest, listener);
}
//StepDetector
@Override
public void onSensorChanged(SensorEvent event) {
//Els canvis només tindran efecte si l'aplicació esta iniciada
        if(started){
                //Quant hi hagui un event del sensor Step detector es mostrara
                if (event.sensor.getType() == Sensor.TYPE_STEP_DETECTOR) {
                        stepCount = stepCount + (int)event.values[0];
                        stepCountV.setText(stepCount+" Passos");
                }
        }
}
```

Llistat de codi 23. Mètodes encarregats de obtenir la informació del GPS i del Step Detector.

6.2.4 Funcionament botons dels diàleg

6.2.4.1 Diàleg guardar ruta

Quant l'usuari decideix finalitzar una ruta, un popupwindow en forma de diàleg apareix, i mostra les opcions de guardar la ruta o descartar-la.



Captura 31. Captura del diàleg guardar ruta.

Al escollir la opció de guardar, abans de tractar les dades obtingudes es realitzen una sèrie de comprovacions: si l'usuari ha escrit un nom al text box, si hi ha connexió, que realment hi haguí una ruta per a guardar, i si el nom que se li vol posar a la nova ruta ja existeix. Amb totes les comprovacions fetes finalment s'iniciarà un thread per comunicar-se amb el backend i guardar la ruta. En canvi al escollir la opció de descartar, l'aplicació es restaurarà per a poder iniciar una captura d'una nova ruta.

```
private void saveDataDialog(){
        //Crear PopUpWindow
        LayoutInflater layoutInflater = (LayoutInflater)
        v.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View popupView = layoutInflater.inflate(R.layout.save_layout, null);
        final PopupWindow popupWindow = new PopupWindow(popupView,LayoutParams.WRAP_CONTENT,
        LayoutParams.WRAP_CONTENT);
        popupWindow.setFocusable(true);
        TextView tv = (TextView) popupView.findViewById(R.id.textSave);
        tv.setText(" Introdueix un nom per guardar la ruta ");
        final EditText name = (EditText) popupView.findViewById(R.id.nameRuta);
        //Guardar ruta
        Button btnGuardar = (Button) popupView.findViewById(R.id.bGuadar);
        btnGuardar.setOnClickListener(new Button.OnClickListener() {
                @SuppressWarnings("unchecked")
                @Override
                public void onClick(View v) {
                        save=true;
                        //Comprovar que hi ha un nom en el textBox
                        if(name.getText().length()==0){
                                Toast.makeText(v.getContext(), "Introdueix un nom per guardar
                                la ruta",Toast.LENGTH_SHORT).show();
                        }else{
                                //Comprovar connexio
                                if(netTools.checkNetwork(v)
                                        mapFragment = (MapFragment) getFragment(1);
                                        listFragment = (ListFragment) getFragment(3);
                                        //Comprovar que hi hagui una ruta per guardar
                                        if(mapFragment.getTrack().size()==0)
                                        {
                                                Toast.makeText(v.getContext(), "No hi ha una
                                                ruta per guardar",Toast.LENGTH_LONG).show();
                                                resetApp();
                                        }else{
                                                //Comprovar que el nom que se li vol posar a
                                                la ruta no existeix
                                                for (int i = 0; i <
                                                listFragment.getDATALIST().size(); i++) {
                                                        if(listFragment.getDATALIST().get(i).
                                                        getName().equals(name.getText().toStr
                                                        ing())){
                                                                save=false;
                                                        }
                                                }
                                                if(save==true){
                                                        //Recollir dades
                                                        pdialog.show();
                                                        new saveDataTask().execute(data);
                                                }else{
                                                        Toast.makeText(v.getContext(),
                                                        "ERROR: ja existeix una ruta amb
                                                        aquest
                                                        nom",Toast.LENGTH_LONG).show();
                                                        Toast.makeText(v.getContext(), "Tria
                                                        un altre nom per la
                                                        ruta.",Toast.LENGTH_LONG).show();
                                                        startPause.setVisibility(View.GONE);
                                                        stop.setText("Guardar ruta");
                                                }
                                        }
                                        }else{
                                                startPause.setVisibility(View.GONE);
                                        stop.setText("Guardar ruta");
                                        }
                                        popupWindow.dismiss();
                                }
                        }
                });
.
.
.
```

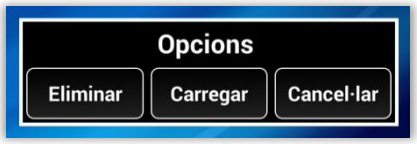
Llistat de codi 24.1. Mètode que mostra el diàleg guardar ruta i controla els botons.

```
.
.
.
        //Descartar ruta
        Button answer = (Button) popupView.findViewById(R.id.bDescartar);
        answer.setOnClickListener(new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                        //Restaurar aplicació
                        resetApp();
                        mapFragment = (MapFragment) getFragment(1);
                        mapFragment.resetMap();
                        popupWindow.dismiss();
                }
        });
        //Mostrar PopUpWindow
        popupWindow.showAtLocation(stop, Gravity.CENTER, 0, 0);
}
```

Llistat de codi 24.2. Mètode que mostra el diàleg guardar ruta i controla els botons.

6.2.4.2 Diàleg consultar ruta

En la pàgina on hi ha la llista de totes les rutes guardades, si fem una pulsació de llarga durada en una d'elles s'obrirà un diàleg amb diferents opcions que afecten a una ruta. Es pot eliminar la ruta, carregar-la a l'aplicació, o cancel·lar, sortir del diàleg.



Captura 32. Captura del diàleg consultar ruta.

La seva posició dins del ListView, i la Llista (DATALIST) on es guarden les rutes localment per millorar la rapidesa en carregar-les, i s'actualitza cada cop que s'actualitza el ListView, es guarda en una variable (pos) esperant les possibles accions del usuari.

Al seleccionar eliminar es busca la ruta a la Llista en la posició (pos), i s'envia al thread per a que la elimini del backend. Mentrestant el thread treballa, es restauren les variables, la Llista (DATALIST), i el adaptador del ListView, perquè quant s'actualitzi la llista més endavant totes les rutes estiguin a la posició correcta.

En cas que vulgui veure la ruta, seleccionarà carregar. En aquest cas l'aplicació es restaurarà per a rebre les noves dades de la ruta seleccionada. Aquestes s'agafaran de la Llista (DATALIST) i es carregaran al layout corresponent.

```
listV.setOnItemLongClickListener(new OnItemLongClickListener() {
        //Listener que capturara el event quant hi hagui un long press a un element de la
        llista
        public boolean onItemLongClick(AdapterView<?> arg0, View arg1,final int pos, final
        long id) {
                //Crear PopUpWindow
                LayoutInflater layoutInflater = (LayoutInflater)
                v.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                View popupView = layoutInflater.inflate(R.layout.long_press_layout, null);
                final PopupWindow popupWindow = new
                PopupWindow(popupView,LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
                popupWindow.setFocusable(true);
                TextView tv = (TextView) popupView.findViewById(R.id.textOpcions);
                tv.setText("Opcions");
                //Eliminar ruta
                Button btnDelete = (Button) popupView.findViewById(R.id.bEliminar);
                btnDelete.setOnClickListener(new Button.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                if(netTools.checkNetwork(v)){
                                        deleteDialog.show();
                                        //Eliminer ruta seleccionada
                                        new deleteDataTask().execute(DATALIST.get(pos));
                                        //Restaurar adaptador, i llista
                                        dataListAdapter.clear();
                                        DATALIST.clear();
                                        String[] loadingListView = {"Actualitzant
                                        rutes....."};
                                        listV.setEnabled(false);
                                        listV.setAdapter(new
                                        ArrayAdapter<String>(v.getContext(),android.R.layout.
                                        simple_list_item_1, loadingListView));
                                        //Actualitzar llista
                                        new getDataTask().execute();
                                }
                                        popupWindow.dismiss();
                        }
                });
                //Carregar ruta
                Button btnLoad = (Button) popupView.findViewById(R.id.bCarregar);
                btnLoad.setOnClickListener(new Button.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                //Restaurar app
                                mapFragment = (MapFragment) getFragment(1);
                                mainFragment = (MainFragment) getFragment(2);
                                mainFragment.resetApp();
                                mapFragment.resetMap();
                                //Carregar dades
                                carregarDades(pos);
                                popupWindow.dismiss();
                        }
                });
                //Cancelar longPress
                Button btnCanel = (Button) popupView.findViewById(R.id.bCancelar);
                btnCanel.setOnClickListener(new Button.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                popupWindow.dismiss();
                        }
                });
                //Mostrar PopUpWindow
                popupWindow.showAtLocation(listV, Gravity.CENTER, 0, 0);
                return true;
        }
});
```

Llistat de codi 25. Control del ListView i dels botons de consultar ruta.

6.2.5 Connexions amb el backend

La millor forma per a executar les accions de guardar, eliminar i llegir les dades en el Cloud Datastore, és utilitzant un AsyncTask .

L'AsyncTask s'utilitza especialment per a tasques de curta durada com les connexions a servidors, permet tenir un control total durant l'execució del thread a traves dels seus mètodes onProgressUpdate(), i onCancelled(), i al finalitzar la tasca a onPostExecute().

6.2.5.1 Guardar dades

Quant totes les dades de la ruta que es vulgui guardar s'hagin comprovat, el següent AsyncTask s'iniciarà. Primer de tot es recuperaran les dades enviades anteriorment, i es guardaran en una instancia de Data per a pujar-la al Cloud Datastore. Utilitzaré el endpoint que he desplegat anteriorment al backend per a realitzar un insert de les dades i al mateix temps, comprovaré que s'han inserit les dades correctament.Al finalitzar es comprova el resultat de l'insert i s'executa el codi corresponent.

```
private class saveDataTask extends AsyncTask<HashMap<String,Object>, Void, String> {
        @Override
        protected String doInBackground(@SuppressWarnings("unchecked")
        HashMap<String,Object>... params) {

                Data data = new com.rungo.dataendpoint.model.Data();
                String result="";

                //Recuperar les dades
                data.setMaxV((Float)params[0].get("max"));
                data.setDistance((Float)params[0].get("meters"));
                data.setAvgV((Float)params[0].get("avg"));
                data.setName(String.valueOf(params[0].get("name")));
                data.setTrack((Text)params[0].get("ruta"));
                data.setTime((Long)params[0].get("time"));
                data.setDate((String)params[0].get("date"));
                data.setPasses((int)params[0].get("steps"));
                data.setUser(preferences.getString("user", null));

                Dataendpoint.Builder builder = new
                Dataendpoint.Builder(AndroidHttp.newCompatibleTransport(), new
                JacksonFactory(), null);

                builder = CloudEndpointUtils.updateBuilder(builder);
                builder.setApplicationName("RunGo");
                Dataendpoint endpoint = builder.build();

                try {
                        //Insertar les dades i comprovar que no hi ha hagut cap error
                        if(endpoint.insertData(data).execute().isEmpty())
                        {
                                result = "empty";
                        }else{
                                result = "ok";
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                        result = e.getMessage();
                }
                return result;
        }
        .
        .
        .
```

Llistat de codi 26.1. AsyncTask que guarda les dades en el Cloud Datastore.

```
        .
        .
        .
        @Override
        protected void onPostExecute(String result) {
                if(result.equals("ok")){
                        //Restaurar l'aplicació
                        startPause.setEnabled(true);
                        Toast.makeText(v.getContext(), "Ruta guardada
                        correctament",Toast.LENGTH_SHORT).show();
                        resetApp();
                        mapFragment = (MapFragment) getFragment(1);
                        mapFragment.resetMap();
                        listFragment = (ListFragment) getFragment(3);
                        listFragment.refeshList();
                }else{
                        //Mostrar error
                        Toast.makeText(v.getContext(), result,Toast.LENGTH_SHORT).show();
                }
                pdialog.cancel();

                getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SE
                NSOR);
        }
}
```

Llistat de codi 26.2. AsyncTask que guarda les dades en el Cloud Datastore.

6.2.5.2 Consultar dades

Aquesta tasca s'executa cada cop que s'inicia l'aplicació i s'ha d'actualitzar la llista de les rutes. Només necessita saber quin usuari està executant l'aplicació per a mostrar les seves rutes.

Per a recollir les dades que em retorna al executar (endpoint.listData().execute), utilitzo una col·lecció de dades Data que em permetrà guardar una llista.

Després d'obtenir les dades, es crea el adaptador personalitzat (TrackListAdapter), que contindrà el nom de la ruta i la data i hora en que s'ha guardat per omplir les dades a la llista, i es guarda una instancia de la llista per a una posterior consulta. Ja que al executar (listData()) retorna totes les rutes disponibles en el Cloud Datastore s'ha de seleccionar les del usuari que ha realitzar la consulta mentrestant es crea el adaptador. En el cas de que no hi hagi cap ruta es crearà un adaptador simple informant de que no hi ha rutes.

```
//Recollir totes les dades guardades
private class getDataTask extends AsyncTask<Void, Void, CollectionResponseData> {
        @Override
        protected CollectionResponseData doInBackground(Void... params) {
                Dataendpoint.Builder builder = new
                Dataendpoint.Builder(AndroidHttp.newCompatibleTransport(), new
                JacksonFactory(), null);
                builder = CloudEndpointUtils.updateBuilder(builder);
                builder.setApplicationName("RunGo");
                Dataendpoint endpoint = builder.build();
                //La resposta sera una llista de dades, List<Data>
                CollectionResponseData result;
                try {
                        result = endpoint.listData().execute();
                } catch (IOException e) {
                        e.printStackTrace();
                        result = null;
                }
                return result;
        }
        @Override
        protected void onPostExecute(CollectionResponseData result) {
                //Comprovar que el resultat no sigui null
                if(result == null)
                {
                        String[] noTracks = {"No hi han rutes guardades"};
                        listV.setAdapter(new
                        ArrayAdapter<String>(v.getContext(),android.R.layout.simple_list_item
                        _1, noTracks));
                        listV.setEnabled(false);
                }else{
                        //Crear l'adapter i carregar-lo si hi han rutes
                        dataListAdapter = new TrackListAdapter(v.getContext(),
                        manageData(result.getItems()));
                        if(DATALIST.size()==0){
                                String[] noTracks = {"No hi han rutes guardades"};
                                listV.setAdapter(new
                                ArrayAdapter<String>(v.getContext(),android.R.layout.simple_l
                                ist_item_1, noTracks));
                                listV.setEnabled(false);
                        }else{
                                listV.setEnabled(true);
                                listV.setAdapter(dataListAdapter);
                        }
                }
        }
        //Recollir les dades del usuari i escriure les rutes si es necessari
        private ArrayList<TrackListItems> manageData(List<Data> allData) {
                ArrayList<TrackListItems> items = new ArrayList<TrackListItems>();
                String user = preferences.getString("user", null);
                try {
                        if(allData != null)
                        {
                                for (Data data : allData) {
                                        //Comprovar que la ruta sigui del usuari actual
                                        if(data.getUser().equals(user)){
                                                DATALIST.add(data);
                                                items.add(new
                                                TrackListItems(data.getName(),data.getDate())
                                                );
                                                //Escriure fitxer a la memoria SD
                                                gpx.write(data.getTrack().getValue()
                                                ,data.getName());
                                        }
                                }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        return items;
        }
}
```

Llistat de codi 27. AsyncTask que obté les dades del Cloud Datastore.

6.2.5.3 Eliminar dades

En el diàleg que s'obra al seleccionar una ruta de la llista hi ha l'opció d'eliminar-la completament. Per eliminar la ruta farà falta tenir la clau identificadora i el nom, els quals s'envien al AsyncTask per paràmetre.
Un cop tenim les dades buscarem el fitxer GPX de la ruta a la memòria del dispositiu mòbil i el carregarem en un fitxer, tot seguit ens comunicarem amb el backend per eliminar la ruta utilitzant el id de la clau i finalment s'eliminarà el fitxer del dispositiu mòbil retornat una confirmació de que s'ha eliminat. Depenent del resultat es mostra un toast informant de èxit de la tasca.

```
//Eliminar dades del cloud i de la memoria SD del dispositiu mobil
private class deleteDataTask extends AsyncTask<Data, Void, Boolean> {
        @Override
        protected Boolean doInBackground(Data... params) {
                //Carregar fitxer de la memoria SD del dispositiu mobil
                File d = new
                File(Constants.FILE_ROUTE+params[0].getName()+Constants.FILE_EXTENSSION);

                Dataendpoint.Builder builder = new
                Dataendpoint.Builder(AndroidHttp.newCompatibleTransport(), new
                JacksonFactory(), null);

                builder = CloudEndpointUtils.updateBuilder(builder);
                builder.setApplicationName("RunGo");
                Dataendpoint endpoint = builder.build();

                try {
                        endpoint.removeData(params[0].getKey().getId()).execute();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                //Eliminar fitxer de la memoria SD del dispositiu mobil
                return d.delete();
        }
        @Override
        protected void onPostExecute(Boolean result) {
                deleteDialog.cancel();
                //Comrovar que el fitxer s'ha eliminat correctament
                if(result == true)
                {
                        Toast.makeText(v.getContext(),"Ruta eliminada correctament" ,
                        Toast.LENGTH_LONG).show();
                }else{
                        Toast.makeText(v.getContext(),"Ruta no eliminade de la memoria
                        interna" , Toast.LENGTH_LONG).show();
                }
                //Restaurar aplicació
                mapFragment = (MapFragment) getFragment(1);
                mainFragment = (MainFragment) getFragment(2);
                mainFragment.resetApp();
                mapFragment.resetMap();
        }
}
```

Llistat de codi 28. AsyncTask que elimina les dades del Cloud Datastore i el fitxer GPX de la ruta.

7 Conclusions

7.1 Conclusions

En resum, aquest treball m'ha servit per aplicar els meus coneixements que he aprés aquests últims anys, ampliar-los, i demostrar que sóc capaç de realitzar un projecte en tots els seus aspectes.

Encara que no ha sigut una tasca fàcil he pogut realitzar totes les funcionalitats que m'havia proposat al iniciar el treball. Alguna m'ha portat més feina del que m'esperava, he tingut de fer algunes recerques en internet per trobar la millor forma de solucionar-ho.

Aplicant algunes de les millores esmentades anteriorment, hem podria pensar de posar l'aplicació en el Market. Així podria experimentar tot el recorregut de desenvolupar una aplicació i fer-la arribar a usuaris que la necessitin.

Tot i que he aprés molt sobre Android durant l'elaboració del projecte crec que me queda molt més per aprendre ja que Android ens brinda moltes possibilitats a l'hora de ser creatius i disposa d'una gran quantitat de eines que ens faciliten el desenvolupament.

En un futur m'agradaria desenvolupar aplicacions per a dispositius mòbils, encara que no seguessin de la plataforma Android, ja que és un sector de la informàtica que està tenint un gran creixement.

Crec que aquest projecte em podria ajudar en entrar en el mon laboral en el sector del desenvolupament d'aplicacions per a dispositius mòbils.

7.2 Possibles millores

Qualsevol aplicació necessita un manteniment, actualització, i aquesta no n'és una excepció. Actualment les seves funcionalitats són suficients però no optimes. En versions futures s'hi podria incloure un menú per a tenir un control total sobre les funcionalitats com per exemple:

- Opció per configurar les velocitats del ritme.
- Opció per configurar els colors del ritme.
- Opció per configurar el Zoom durant la ruta.
- Opció per configurar el tipus de mapa que es visualitza.

A més a més estaria be incloure-hi un manual per a la utilització de l'aplicació.

Una altra millora seria la traducció en diferents idiomes per a que qualsevol usuari del mon, contribuint així a la seva internacionalització.

Introduir-hi eines per compartir les rutes en les xarxes socials com Facebook, Twiter, etc.

Proporcionar estadístiques del rendiment i progrés dels usuaris utilitzant els paràmetres obtinguts en totes les rutes. Això pot incloure la visualització d'aquesta informació en forma de gràfiques, i suggeriments per a millorar el rendiment.

Incloure l'aplicació en el Market d'Android no seria una millora, però permetria distribuir-la globalment i obtenir alguns ingressos.

Errors a millorar.

- Gestió de les rutes dels usuaris al App Engine. Actualment en el dispositiu mòbil.
- Gestió dels AsyncTask al rotar la pantalla.

8 Bibliografia

[1] https://developers.google.com/appengine/?hl=es

[2] https://cloud.google.com/developers/articles/how-to-build-mobile-app-with-app-engine-backend-tutorial

[3] https://developers.google.com/kml/documentation/mapsSupport?hl=es

[4] http://danielme.com/category/android/

[5] http://viewpagerindicator.com/

[6] http://www.sgoliver.net/blog/?page_id=3011

[7] http://gpmess.com/blog/category/desarrolladores/#.UzwNpldvAk1

[8] http://stackoverflow.com/

[9] https://developer.android.com/index.html

[10] https://gist.github.com/kamituel/6465125

[11] http://www.topografix.com/gpx.asp

[12] http://blog.tomtasche.at/2013/05/google-oauth-on-android-using.html

[13] http://hmkcode.com/android-custom-listview-items-row/

[14] http://source.android.com/source/licenses.html

[15] http://www.apache.org/licenses/LICENSE-2.0.html

[16] http://www.softicons.com/system-icons/iicons-by-treetog-artwork/run-icon

[17] https://developers.google.com/cloud/terms/?hl=es

[18] https://developer.apple.com/programs/ios/

[19] http://nodejs.org/

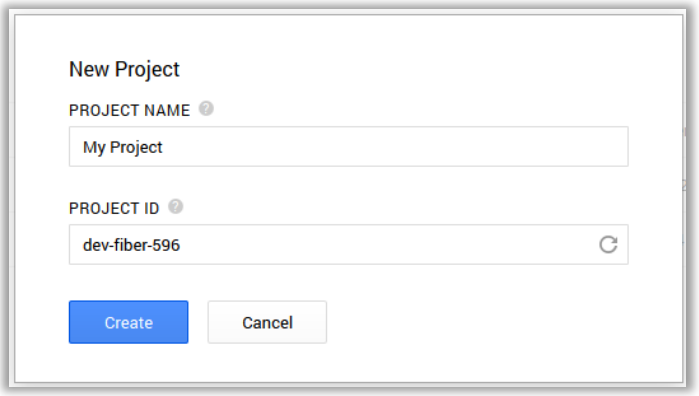
The Android Developer's Cookbook: Building Applications with the Android SDK. J. Steele. Addison-Wesley Professional (2011)

Android in Action. W. F. Ableson, R. Sen, C. King and C. E. Ortiz. Manning Publications Co. Third Edition (2011).

9 Annex - configuracions

En aquest apartat es mostra com realitzar les configuracions necessàries per la utilització de les eines de Google.

Primer de tot, un cop s'ha accedit a la consola necessitarem crear un nou projecte per a l'aplicació. S'haurà de triar un nom per al projecte i un identificador.

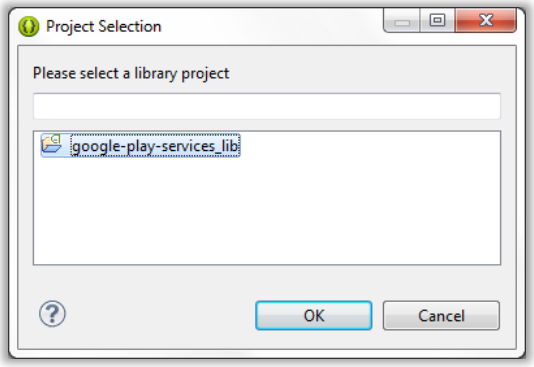


Captura 33. Captura Consola Google de la creació d'un projecte.

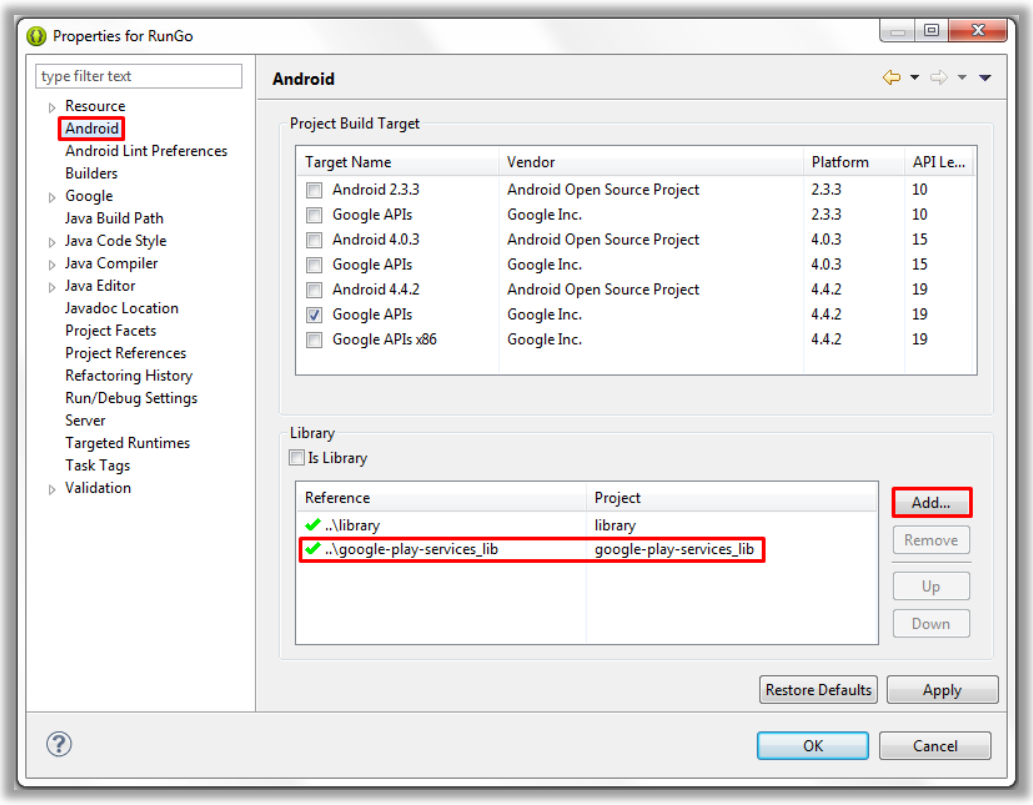
9.1 Configuració Google Maps

Per a poder utilitzar el Google Maps [6], es te de realitzar una sèrie de configuracions prèvies.

S'ha d'importar la llibreria de google play services, la qual també ens servirà per utilitzar l'App engine. Aquesta llibreria es troba en el SKD d'Android, i es te d'importar al workspace on hi ha el projecte el qual la volem utilitzar. Un cop importada accedim a les propietats del projecte i afegim la llibreria tal com es mostra en les següents captures.

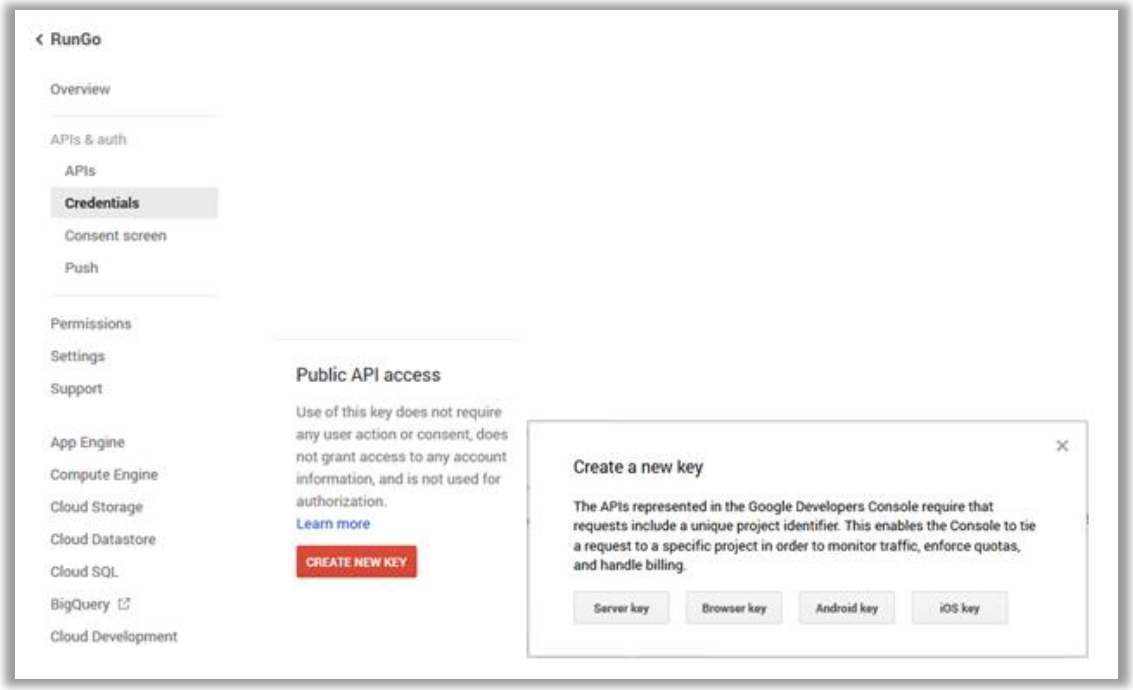


Captura 34. Captura Eclipse de la finestra per importat una llibreria.



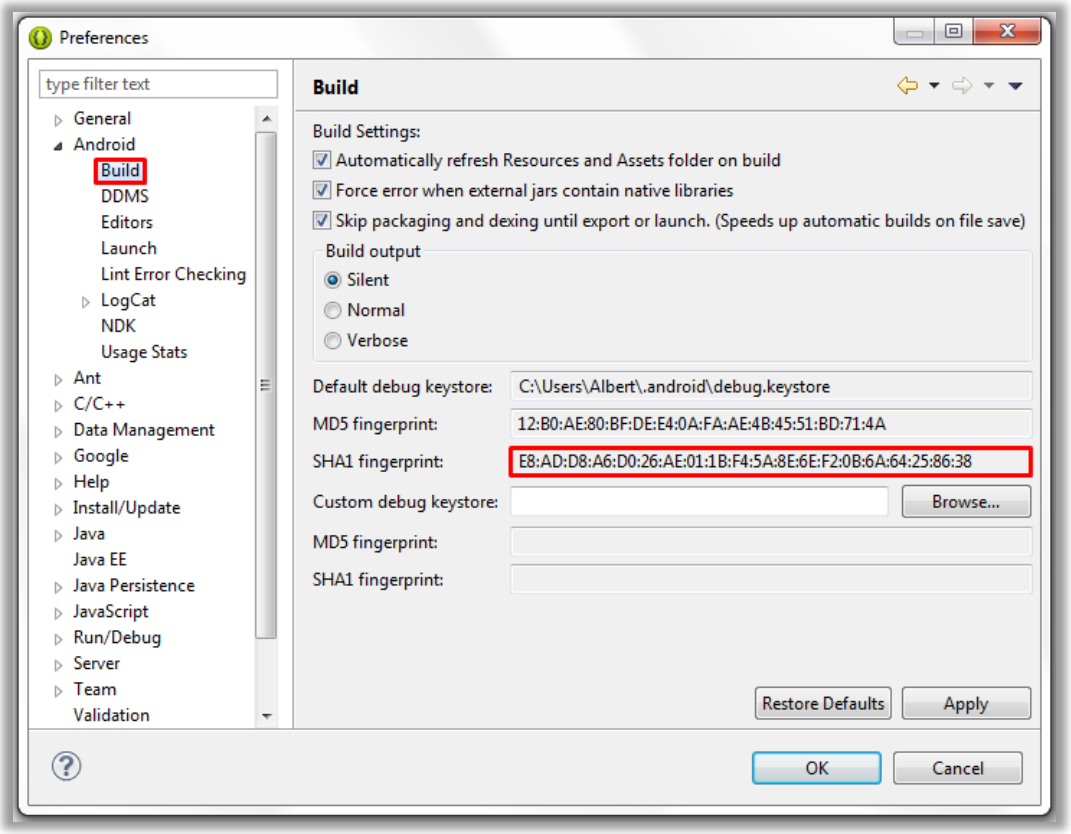
Captura 35. Captura Eclipse de les propietats del projecte de l'aplicació.

Necessitarem una Api Key per a poder utilitzar el servei de mapes de Google. Per a crear-la s'ha d'accedir a la Consola de APIs de Google -> Projecte -> APIs & auth -> Credentials -> CREATE NEW KEY -> Android key.



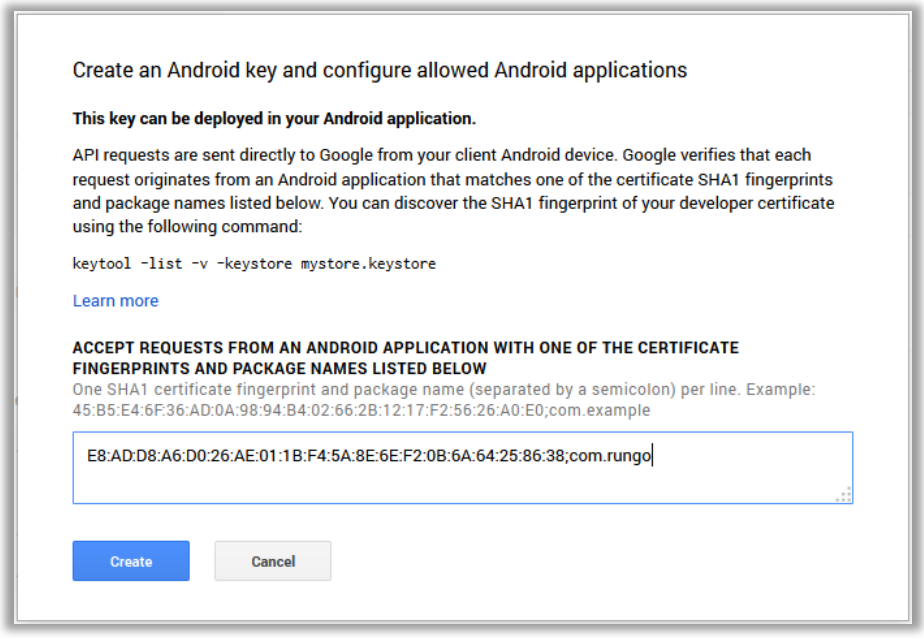
Captura 36. Captura Consola Google de la creació d'una key.

Un cop s'obra la següent finestra ens demana que obtinguem el SHA1 fingerprint del nostre entorn de desenvolupament. Es pot obtenir fàcilment en l'Eclipse -> Window -> Preferences -> Android -> Build.

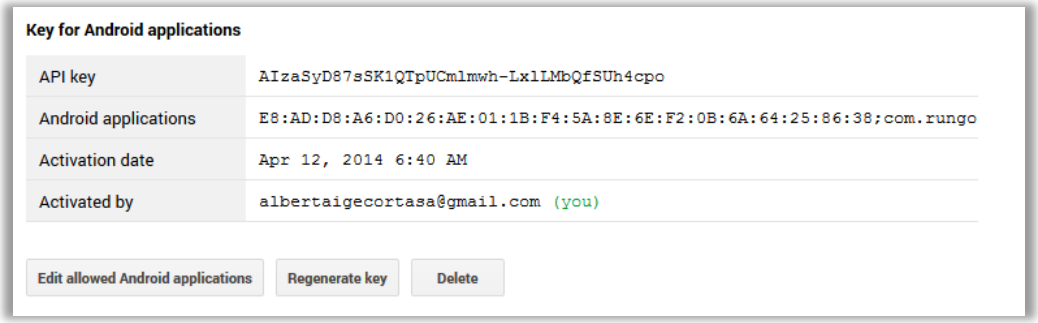


Captura 37. Captura Eclipse del SHA1 del entorn de desenvolupament.

Per acabar s'ha de copiar i pegar el SHA1 fingerprint en el requadre i posar-hi ; al final, tot seguit s'ha de posar el nom del paquet del projecte.



Captura 38. Captura Consola Google de la creació de l'Android key.



Captura 38. Captura Consola Google de la API key per utilitzar en el Google Maps.

Finalment a la Consola de APIs de Google farà falta activar el Google Maps Android API v2, que es troba a APIs & auth -> APIs-

Captura 39. Captura Consola Google de la versió de Google Maps necessària.

Ara només falta anar al manifest del projecte i afegir-hi uns permisos, la versió d'OpenGL ES version 2 que utilitza la API v2 de Google Maps Android, i la nostra api key.

```
<manifest>
        .
        .
        <uses-permission android:name="android.permission.INTERNET"/>
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
        <uses-permission
        android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
        <uses-feature android:glEsVersion="0x00020000" android:required="true"/>
                .
                .
        <application>
                .
                .
        <meta-data android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyD87sSK1QTpUCmlmwh-LxlLMbQfSUh4cpo"/>
                .
                .
        </application>
        .
        .
</manifest>
```

Llistat de codi 29. Permisos i versió d'OpenGL necessaris en el manifest de l'aplicació.

En el layout del mapa s'ha d'afegir el següent codi. A més a més he inclòs un marge al mapa per a millorar el desplaçament entre pàgines.

```
<LinearLayout>
    <com.google.android.gms.maps.MapView
                android:id="@+id/map"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_margin="5dp"/>
</LinearLayout>
```

Llistat de codi 30. Widget de Google Maps en el layout.

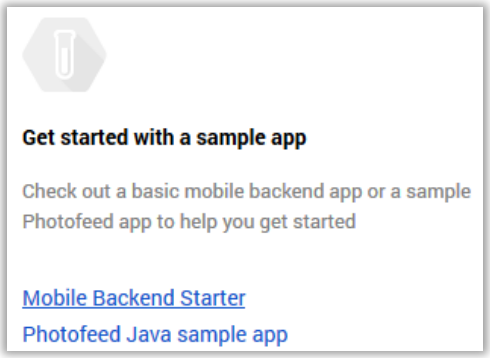
Al iniciar el fragment que conte el mapa, configuro el mapa al meu gust per un funcionament correcte.

```
private void initMap(){
        //Inicialitzar mapa abans de fer cap crida a la camera
        try {
                MapsInitializer.initialize(v.getContext());
        } catch (Exception e) {
                e.printStackTrace();
        }
        //Configurar mapa
        map = mapView.getMap();
        map.getUiSettings().setMyLocationButtonEnabled(true);
        map.getUiSettings().setRotateGesturesEnabled(true);
        map.setMyLocationEnabled(true);
        map.getUiSettings().setCompassEnabled(true);
        map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
}
```

Llistat de codi 31. Mètode que inicia el mapa.

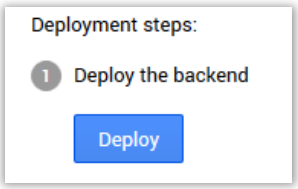
9.2 Configuració App engine

En la pantalla principal del projecte que s'ha creat abans, es pot veure un enllaç al Mobile Backend Starter [2], el qual ens porta a configurar el App engine.



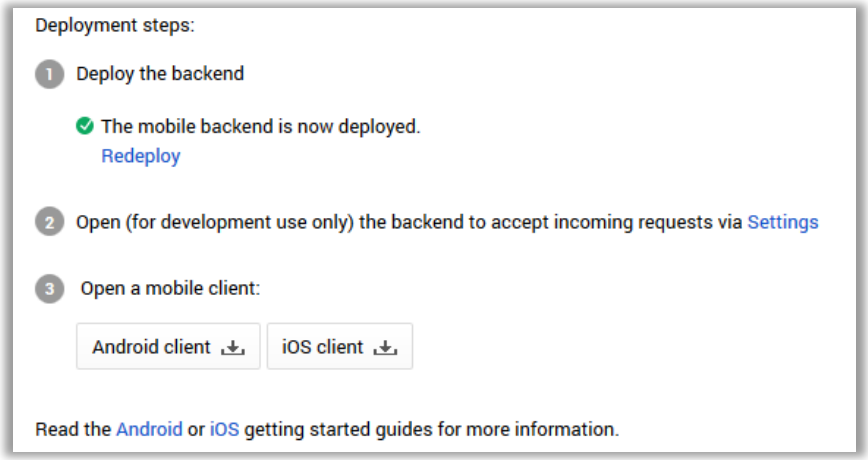
Captura 40. Captura Consola Google del Mobile Backend Starter.

Un cop dins s'ha d'iniciar el desplegament del App engine al projecte. Segons la connexió a internet pot tardar més o menys.



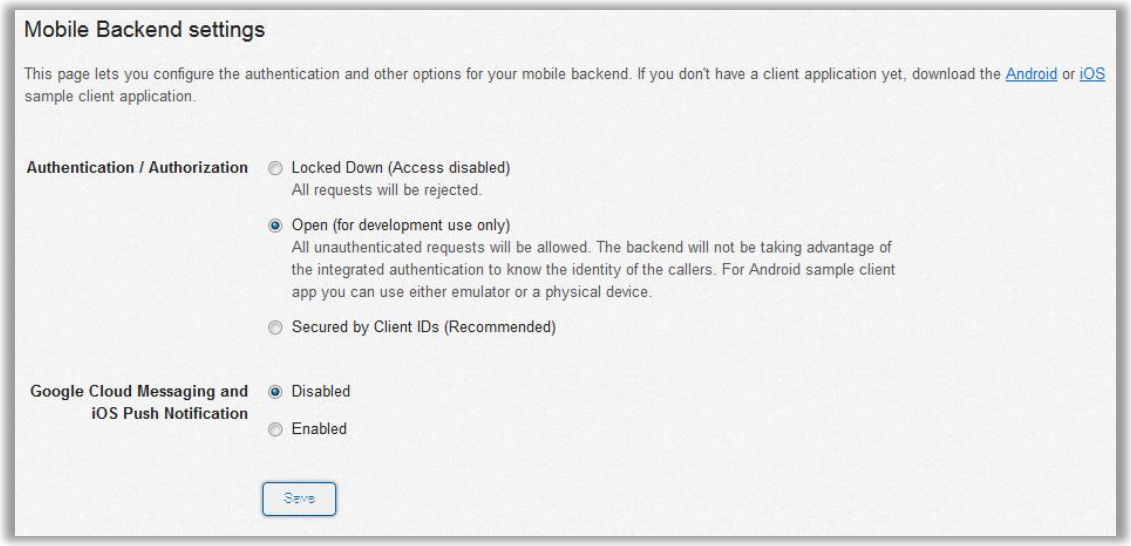
Captura 41. Captura Consola Google del desplegament del backend en el projecte.

El següent pas serà obrir els permisos per accedir al backend, clicant a Settings.



Captura 42. Captura Consola Google dels passos per desplegar el backend.

Un cop a dins ens trobarem aquestes opcions entre d'altres. L'Authentication de moment la posarem a Open per començar a desenvolupar l'aplicació, més tard quan s'implementi l'autenticació s'haurà de configurar una sèrie de paràmetres en Secured by Client IDs. I com que l'aplicació no necessita rebre missatges push ho deixem en Disabled.



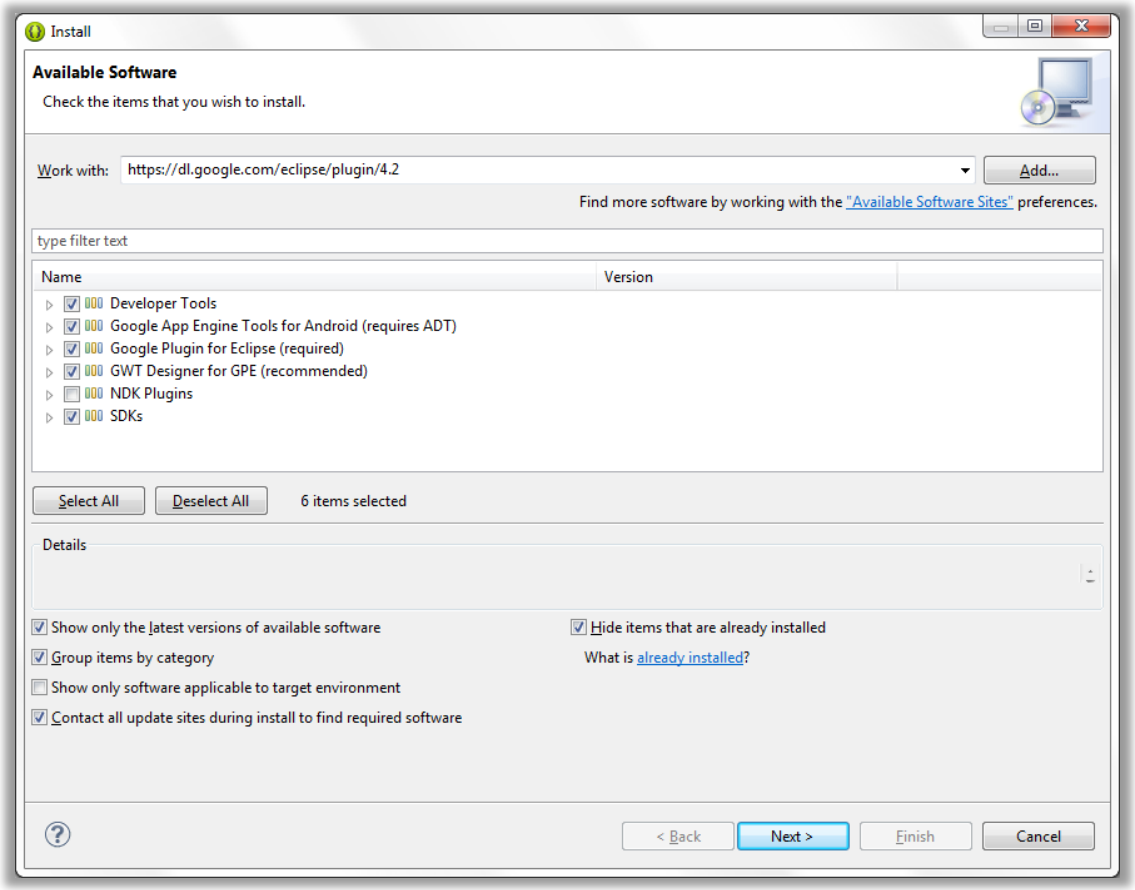
Captura 43. Captura de les configuracions del Mobile Backend.

En el 3r pas et pots descarregar un exemple per a provar les funcionalitats del backend, que no és necessari.

De moment fins aquí no s'ha de configurar res més del App engine. Ara s'ha de configurar el App engine en el entorn de desenvolupament, en el meu cas l'Eclipse, per a poder crear les classes base per a crear les Entitys, Endpoints, i desplegar una versió de l'aplicació al backend per a comunicar l'aplicació amb l'App engine.

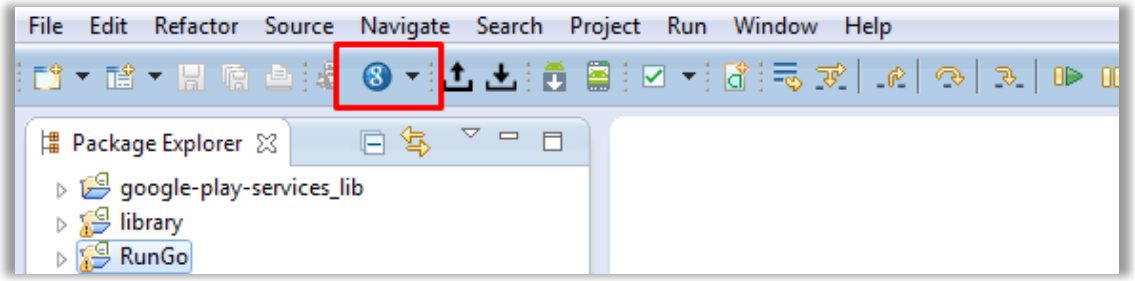
Dins del Eclipse es selecciona Help -> Install new software. En el diàleg s'hi posa la següent url i un cop surten els paquets a instal·lar els seleccionem tots menys el NDK Plugins. Un cop finalitzada la instal·lació l'Eclipse demanarà reiniciar-se.

https://dl.google.com/eclipse/plugin/4.2



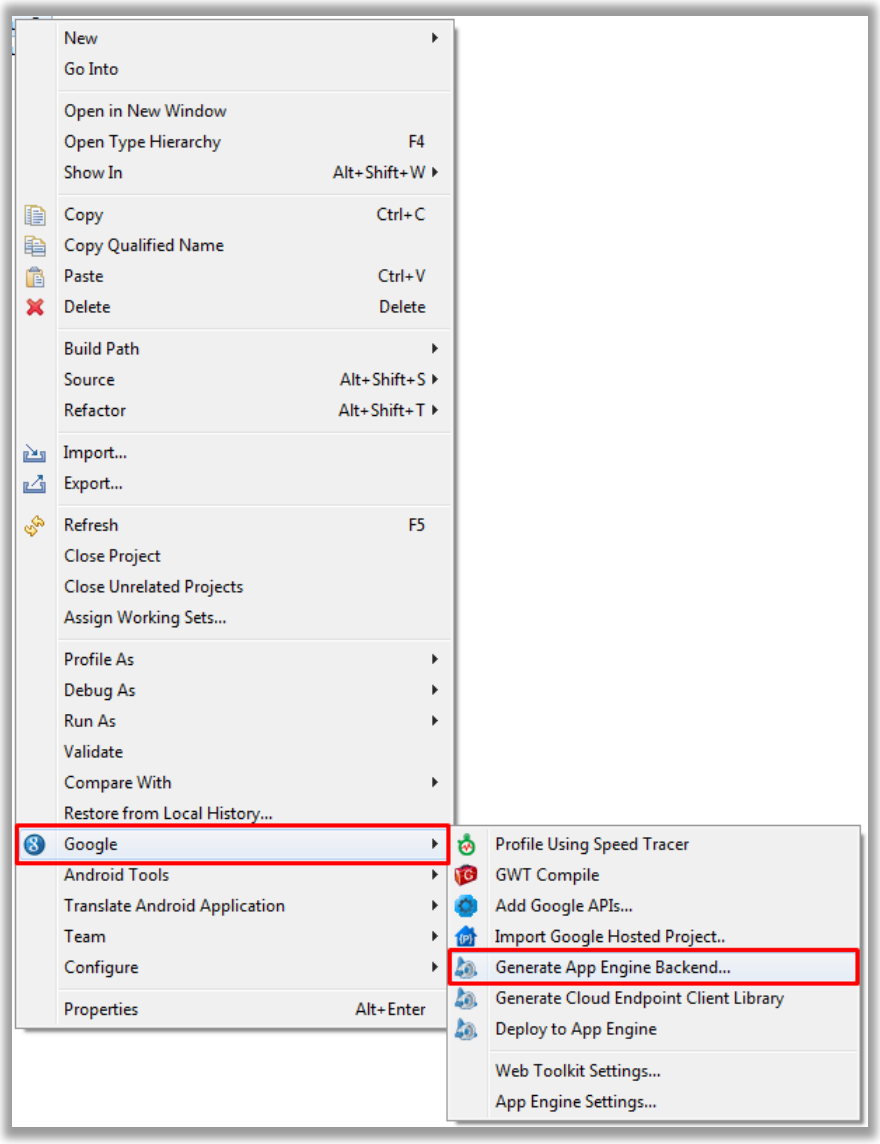
Captura 44. Captura Eclipse per instal·lar plugins/nou software.

Un cop l'Eclipse s'hagi reiniciat veurem el pluguin a la barra d'eines superior.



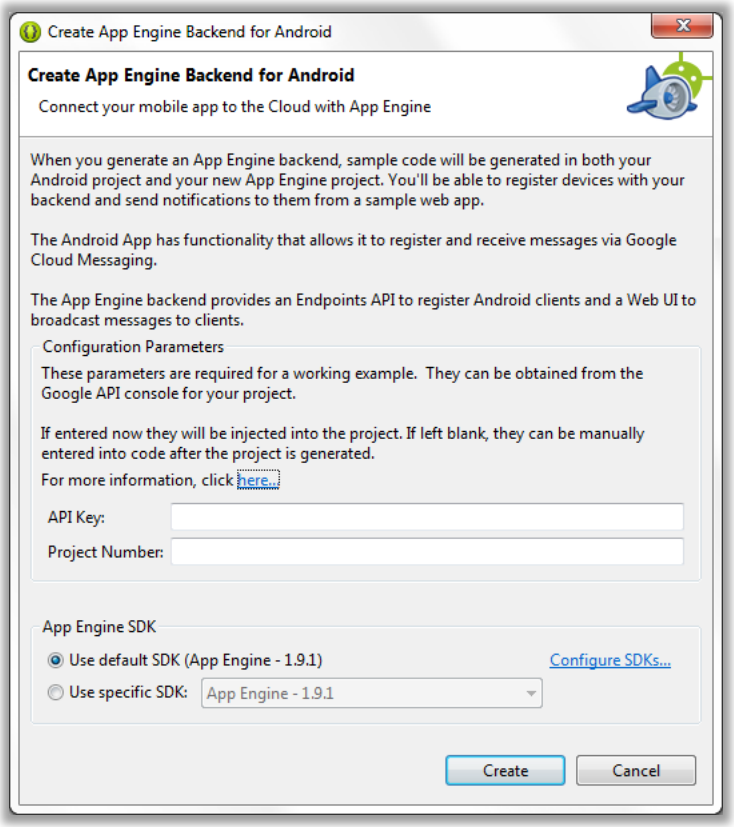
Captura 45. Captura Eclipse del plugin AppEngine.

Per afegir i treballar el App engine en l'aplicació, tindrem de generar el App engine backend per el projecte de l'aplicació. Fent clic dret sobre el projecte, en el desplegable anar a Google -> Generate App Engine Backend.



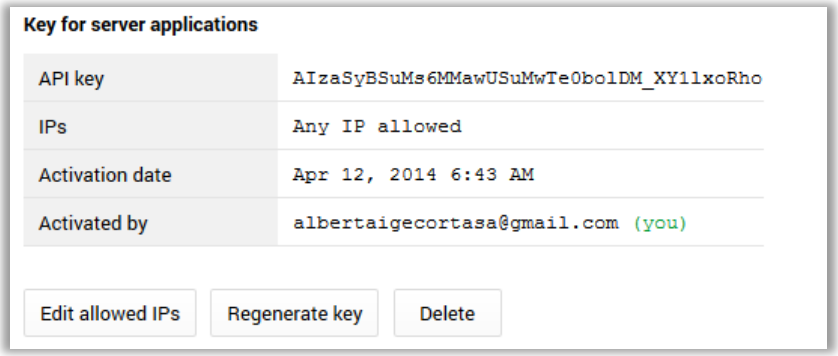
Captura 46. Captura Eclipse per crear el projecte App Engine de l'aplicació.

Sortirà una finestra on demana una API key, el Project Number, i ens deixa triar la versió del SDK del App engine, el qual ho deixarem com esta.



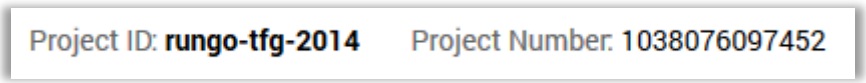
Captura 47. Captura Eclipse creació del projecte App Engine per l'aplicació.

La API key que demana serveix per utilitzar el Google Cloud Messaging que no faré servir.



Captura 48. Captura Consola Google API key per aplicacions del servidor.

I el Project Number el trobem a la pàgina inicial del projecte, a la part superior esquerra al costat del Project ID.



Captura 49. Caputra Consola Google dels identificadors del projecte.